

---

フロントページの続き

(71)出願人 Stirling House, Stirling Road The Surrey Research Park Guildford, Surrey GU2 5RF England

(72)発明者 スティーリング アンドリュー ジェームス  
イギリス サーレイ ケイティー17 3エルピー エプソン ダウンズ ルーデンウェイ 68

F ターム(参考) 5K031 DA02 DA19 EA12 EC01 EC05  
5K034 AA20 DD03 EE02 EE11 JJ21

7

# DEVELOPMENT AND TEST TOOLS FOR COMMUNICATION SYSTEM

Publication number: JP2003526223 (T)

Publication date: 2003-09-02

Inventor(s):

Applicant(s):

Classification:

- International: G06F11/28; G06F13/00; G06F13/14; H04L12/24; H04L12/26; H04L29/06; H04L29/14; G06F11/28; G06F13/00; G06F13/14; H04L12/24; H04L12/26; H04L29/06; H04L29/14; (IPC1-7): G06F11/28; G06F13/00; G06F13/14; H04L29/06; H04L29/14

- European: H04L12/24F1; H04L12/26T

Application number: JP20000512312T 19980914

Priority number(s): GB19970019412 19970912; GB19970019403 19970912; WO1998GB02721 19980914

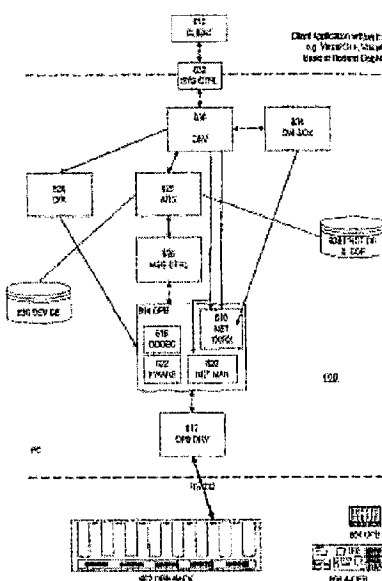
Also published as:

WO9914891 (A2)  
WO9914891 (A3)  
EP1013033 (A2)

Abstract not available for JP 2003526223 (T)

Abstract of corresponding document: **WO 9914891 (A2)**

The development system comprises reusable computer implemented components to be used in conjunction with a hardware network interface board, for emulation of devices in a functioning network. The same components can provide tools for stimulation and analysis of network traffic and compliance verification. The components provide an interface between a client program and the network interface hardware. They include a protocol database, in which application-level protocols specific to devices such as CD changers, amplifiers and audio-video control units are defined. The proposed tools offer a mechanism for filtering commands instructed by the user (client program), where these conflict with application protocols defined for the specific type of device being emulated.; The components include program modules for automatic implementation of standard behaviours, such as network management and source data connection routing. In addition to application protocols generic to all network configurations employing a certain protocol, the tools may store a network system description defining the particular number and type of devices present or emulated on the network, to restrict further the range of legal commands.



Data supplied from the **espacenet** database — Worldwide

(11)特許出願公表番号  
特表2003-526223  
(P2003-526223A)

(51)Int.Cl. <sup>7</sup>	識別記号	F I	テマコード* (参考)
H 0 4 L 29/06		G 0 6 F 11/28	3 4 0 C 5 B 0 1 4
G 0 6 F 11/28	3 4 0	13/00	3 5 3 V 5 B 0 4 2
13/00	3 5 3	13/14	3 3 0 Z 5 B 0 8 9
13/14	3 3 0	H 0 4 L 13/00	3 0 5 Z 5 K 0 3 4
H 0 4 L 29/14			3 1 5 Z 5 K 0 3 5
		審査請求 未請求	予備審査請求 有 (全 64 頁)

(71)出願人 コミュニケーション アンド コントロー  
 ル エレクトロニクス リミテッド  
 COMMUNICATION & CON  
 TROL ELECTRONICS LI  
 MITED  
 英国 ジーユー２ ５ワイキュー サリー  
 ギルドフォード ザ サリー リサーチ  
 パーク オッカム ロード オッカム  
 コート ２

(74)代理人 弁理士 谷 義一 (外2名)

**【特許請求の範囲】**

【請求項1】 開発中の特定の製品をエミュレートするために機能している通信ネットワークに接続されたハードウェアネットワークインタフェースとともに使用されるコンピュータに実装された構成要素を含むシステムであって、前記構成要素が、クライアントプログラムとネットワーク上の他のデバイスにメッセージを送信するためのネットワークインタフェースとの間のインタフェースを提供し、プロトコルデータベースを含み、ネットワーク上に送信するよう命令されている違法メッセージを検出するために、前記構成要素のうちの少なくとも1つがクライアントプログラムから命令されたメッセージをプロトコルデータベースを参照することによって処理することを特徴とするシステム。

【請求項2】 請求項1に記載のシステムであって、前記プロトコルデータベースにより、ネットワーク中のデバイスの機能に特有のアプリケーションレベルのプロトコルを規定し検査することを特徴とするシステム。

【請求項3】 請求項2に記載のシステムであって、前記プロトコルデータベースにより、エミュレートされているデバイスの特定の型と矛盾するためにネットワーク中の様々なデバイスの様々な機能に特有の様々なアプリケーションレベルのプロトコルを規定し検査することを特徴とするシステム。

【請求項4】 請求項3に記載のシステムであって、完成した製品としてまたはエミュレーションによってネットワーク中に存在する他のデバイスの型または機能を特定するシステム記述をさらに含み、ネットワーク上に送信するよう命令されている不適切なメッセージを検出するために前記構成要素の少なくとも1つがクライアントから命令されたメッセージを前記システム記述を参照して処理することを特徴とするシステム。

【請求項5】 請求項1ないし4のいずれかに記載のシステムであって、ネットワーク中の複数のステーション間の拡張された協調動作が所望の結果を達成するために必要である場合に、プロトコルが少なくとも1つの挙動のシーケンスを規定することを特徴とするシステム。

【請求項6】 請求項5に記載のシステムであって、特定のデバイスの機能とは無関係に、前記シーケンスが、ネットワーク管理のための少なくとも1つの

シーケンスを含むことを特徴とするシステム。

【請求項7】 請求項5または6に記載のシステムであって、前記シーケンスが、前記メッセージに加えてネットワークを介して伝達されるデータストリームの経路指定を確立するための少なくとも1つの挙動シーケンスを含むことを特徴とするシステム。

【請求項8】 請求項5、6、または7に記載のシステムであって、クライアントプログラムとの関与なしに、前記挙動シーケンスを自動的に実装する特定の構成要素を含むことを特徴とするシステム。

【請求項9】 請求項8に記載のシステムであって、ネットワークから受信した全てのメッセージが各特定の構成要素に渡され、さらにその特定の構成要素により実装される挙動シーケンスと関連がない場合にはクライアントプログラムに渡されることを特徴とするシステム。

【請求項10】 請求項1ないし9のいずれかに記載のシステムであって、前記構成要素が、所定の規則に従って互いにメッセージを交換する、プログラミングオブジェクトの形式であることを特徴とするシステム。

【請求項11】 請求項1ないし10のいずれかに記載のシステムであって、ある構成要素が同一のコンピュータハードウェア内で動作する様々なアプリケーションの間で共用することができ、他の構成要素がひとつの動作するアプリケーションによって予約されることを特徴とするシステム。

【請求項12】 請求項11に記載のシステムであって、共用される構成要素が、特定のネットワークインタフェースハードウェアとのインタフェースを提供する構成要素を含むことができることを特徴とするシステム。

【請求項13】 請求項1ないし12のいずれかに記載のシステムであって、検出された違法メッセージを送信するか阻止するかを決定するために動作レベルを設定することができることを特徴とするシステム。

【請求項14】 請求項1ないし13のいずれかに記載のシステムであって、検出された違法メッセージを報告するか否か、例えばクライアントプログラムに報告するか否か、決定するために報告レベルを設定することができることを特徴とするシステム。

## 【発明の詳細な説明】

## 【0001】

本発明は、ローカル通信システム中で使用され、開発およびテスト用装置で使用するハードウェアまたはソフトウェアに実装される開発ツールに関する。本発明は特に、例えば消費者の音声／ビデオネットワーク中で、低レベルの通信プロトコルだけでなく、より高レベルなアプリケーションプロトコルにも実装される製品の開発に関する。

## 【0002】

低コストのファイバネットワーク中でソースデータ（CD音声やMPEGビデオ、電話音声など）を制御メッセージと結合するローカル通信システムは、D2B Opticalの形式で提案されている。詳細については、例えばCommunication & Control Electronics Limited、2 Occam Court、Occam Road、The Surrey Research Park、Guildford、Surrey、GU2 5YQ (<http://www.candc.co.uk>) から入手可能な、「Conan Technology Brochure」および「Conan IC Data Sheet」を参照されたい。また、Becker GmbHのヨーロッパ特許出願EP-A-0725516（95P03）、EP-A-0725518（95P04）、EP-A-07225515（95P05）、EP-A-0725520（95P06）、EP-A-0725521（95P07）、EP-A-0725522（95P08）、EP-A-0725517（95P09）、およびEP-A-0725519（95P10）も参照されたい。「Conan」はCommunication & Control Electronics Limitedの登録商標である。本記述はさらに、本発明者等の同時係属の出願PCT/GB98/00349（62793WO）に記載の、倍速インタフェース（double speed interface）も参照する。

## 【0003】

通常はソフトウェアとインタフェースハードウェアの組合せを含むツールとし

ては、一般にパーソナルコンピュータ（P C）または、このようなネットワーク中で使用するために開発中の製品をエミュレートすることができ、既に開発されている現実の製品を検証／テストすることができるその他のデバイスが知られている。本発明は、このような開発ツールの有用性を改善することを目的とするものである。適用分野としては、ユーザインタフェースおよび高レベル制御機構も含めた新製品の製造前のシミュレーション、および現実の製品の検証／互換性テストがある。開発中のシステムの性質として、異なるタイプのいくつかのユニットが同時に開発されることがあり、またそれらの機能要件の定義およびデバッグが非常に複雑になる。特に、ネットワーク中のあるデバイスの故障が、別のデバイスで望ましくないふるまいとして現れ、診断を困難にすることがある。

#### 【0004】

本発明は、第1の態様で、添付の請求項1で規定するシステムを提供する。別の態様では、本発明は、ハードウェアネットワークインタフェースとともに使用されるコンピュータに実装された構成要素を含むツールを提供し、前記ツールの構成要素は、ユーザまたはクライアントプログラムとネットワークインタフェースとの間のインタフェースを規定する。このツールは、少なくとも1つのアプリケーションプログラムインタフェース（A P I）と、プロトコルデータベースおよび／またはシステムモデルと、このデータベース／モデルを参照することによってユーザまたはクライアントプログラムからのメッセージおよびイベントを処理する構成要素とを組み込む。ネットワークに送信されたまたはネットワークから受信した違法メッセージを検出するためである。

#### 【0005】

特に、C Dチェンジャ、アンプ、音声／ビデオ制御装置などのデバイスに特有のアプリケーションレベルのプロトコルは、D 2 B O p t i c a l システムとして定義されている。提案するツールは、ユーザ（クライアントプログラム）から命令されたコマンドを、フィルタリングする機構を提供する。ここではエミュレートされた特定タイプのデバイスについて規定されたプロトコルと矛盾する場合である。一般に、アプリケーションプロトコルは、単なる通信管理またはネットワーク管理のタスクと関係するのではなく、ネットワークに接続されたデバイ

スの本質的な機能と関係するプロトコルである。

【0006】

一般に、プロトコルは、特定の製品設計者の必要に応じて、彼または彼女自身の設計した挙動を達成するために個別のメッセージおよび応答のフォーマットを規定することができるが、プロトコルには挙動のシーケンスを規定する領域が存在し、特にネットワーク中の複数のステーション間の拡張された協調動作が必要な場合に、設計結果を達成する必要がある。例としては、ネットワーク管理（初期化）、および少なくともD2B Opticalにおいて、ソースデータの接続管理がある。このツールは、これらの標準的な挙動を自動的に実施するためのプログラムモジュールを含むことができるために、ツールのユーザが自分のアプリケーションの標準部分を明示的にコーディングする必要はない。

【0007】

本明細書に与える実施形態では、このツールを含むプログラムモジュールは、所定の規則に従って互いにメッセージを交換するオブジェクトの形をとる。いくつかのオブジェクトは、同一コンピュータハードウェアの中で動作している異なるアプリケーション間で共用することができ、その他のオブジェクトは単一の動作中のアプリケーションのために予約される。したがって、開発システムは、異なるクライアントアプリケーションを提供する、予約された各オブジェクトクラスのいくつかのインスタンスを含むことができる。共用されるオブジェクトは、特に、ツールシステム（ハードウェアドライバ）の特定のインタフェースハードウェアに対するインタフェースを提供するオブジェクトを含むことができる。特定の「システムインスタンス」に確保されるオブジェクトは、複数のネットワークデバイスを単一のコンピュータで、完全に独立してエミュレート、テスト、または制御することができる。

【0008】

特定のプロトコルを利用する全てのネットワーク構成に汎用なアプリケーションプロトコルに加えて、ツールは、特定の数およびネットワーク上のまたはエミュレートされるデバイスのタイプを規定するネットワークシステムの記述を記憶し、正常コマンドの範囲をさらに限定することができる。

## 【0009】

本発明はさらに、添付の図面に関連しておよそ本明細書に記述する、製品をエミュレートおよびテストする方法も提供する。このシステムのプログラムおよびハードウェア構成要素は、合わせて提供することも別々に提供することもでき、また既存の汎用コンピュータとともに使用される構成要素として提供することもできることを理解されたい。

## 【0010】

(好適実施形態の詳細な説明)

(D2B Opticalシステムの概要)

例示のみを目的として、D2B Opticalネットワークに適用する本発明の様々な態様について説明する。背景として、このネットワークの性質および動作について最初に簡単に述べる。

## 【0011】

図1に示すシステムは、ローカルエリアネットワーク(LAN)のステーション(またはノード)として接続された、音声または画像と関連する9個の装置101~109を含む。言うまでもなく、9個より多いまたは少ないステーションを収容することもできる。この例のシステムでは、これらの装置は、制御および表示ユニット101、コンパクトディスクメモリ(CD-ROM)読取り装置102、ラジオチューナ103、CDチェンジャユニット104、オーディオパワーアンプ105、ファクシミリ送受信ユニット(FAX)106、画像記録システム(VCR/CAMCORDER)107、ビデオチューナ108、および電話109である。制御および表示ユニット101の表示機能は、例えば、CD-ROMによるメモリデバイスから読取った情報の表示、および/あるいはチューナ108またはVCR107からのビデオ信号の表示を行うことができる。

## 【0012】

既知のシステム中のLANの相互接続は、9本の一方方向性ポイントツーポイント光ファイバリンク111、112など、および連結インタフェースモジュール121などを含み、これらはそれぞれ構造的にほぼ同じであるので、全てのノードが環状に接続されるようになっている。各光ファイバリンクは、以下で詳細に

述べるフレーム構造に従って、デジタル音声／画像信号、CD-ROMデータ、および制御メッセージの組合せを搬送する。制御／表示ユニット101など、指定されたステーション（以下システムマスタと呼ぶ）は、20～50kHz（CDの標本化の場合は通常44.1kHz）のフレームサンプリングレートで連続的にフレーム構造を生成する。スタートアップ時にネットワーク上のあるステーションがシステムマスタとして動作するように指定されるが、システムマスタの役割は、その後、例えば後述のような障害状態にあるときに、別のステーションに再割当することができる。

#### 【0013】

ステーションの光ファイバリングとのインタフェースの実施態様を、図2に概念的に示す。リング119～111から、メディアアクセス制御（MAC）／物理層300が、制御メッセージのための通信管理層302とともにインタフェースモジュール121中に設けられる。通信管理層302は、アドレスの初期化および検証を管理し、規定されたタイミング規則に従って再送することによって、信頼性の高いメッセージの送達を保証する。ソースデータ304のデータ処理および制御メッセージ306は、アプリケーションプロトコルとともにステーションレベル101で提供される。このアプリケーションプロトコルは、通常は、そのステーションのデバイス／サブデバイスのグループ化および制御階層と、製品間で交換される情報のフォーマットと、デバイス／サブデバイスの挙動と、アプリケーションレベルのタイミングとを規定する。インタフェースモジュール121は、例えばConan集積回路、または同様のネットワークランシーバおよびそれと関連する制御ソフトウェアの形で、物理的にステーション内に存在することができることは容易に理解されるであろう。

#### 【0014】

図3に示すように、同様のインタフェースモジュール123は、ラジオカセットプレーヤー103の中の1つの機能として提供され、アンプ310、チューナ312、テープ再生デッキ314、音声／画像制御装置（AVC）316、およびユーザI/O318の機能も有する。これらの機能およびそれらの相互接続は本発明とは直接の関係がなく、図示していない。それらの実施態様は、当業者に

は容易に分かるであろう。

#### 【0015】

図4は、ステーションを既知の光ファイバリングに連結するインタフェースモジュール121（この場合にはノード121）を示す概略図である。LANに接続された全てのステーションは、ソースデータ（3つ以下のチャンネルSD0～SD2）および制御データ（CTRL）を生成し、かつ／または受信することができる。制御データは小容量であり、バースト的に到着し、かつユーザ／イベント駆動（例えばユーザ命令や状態変化など）であるが、ソースデータは連続した大容量のストリーム（例えば音声や圧縮画像、CD-ROMデータなど）である。

#### 【0016】

D2B Opticalシステムでは、ソースデータおよび制御メッセージは、システムマスタとして指定されたステーションが生成したフレームで、ネットワーク上でノードからノードに伝達される。フレームは、音声標本化周波数、通常は $f_s = 44.1 \text{ kHz}$ と同じ速度で循環する。フレームは、48個のフレームからなるブロックにグループ化される。

#### 【0017】

図5は、各ネットワークフレームがどのようにして2つのサブフレーム（「左側」および「右側」）に分割されるかを示す図である。 $f_s = 44.1 \text{ kHz}$ では、毎秒88,200個のサブフレームが存在することになる。左側のサブフレームは、常にネットワーク上を伝送される対の先頭となる。物理レベルでは、ビットは2相符号化されて伝達される。ブロック、フレーム、サブフレーム、および制御フレームの間の関係を図5に示す。

#### 【0018】

図6は、各サブフレームが、トランシーバ内で8バイトフィールドとして処理される64個のビットを、どのように含むかを示す図である。フィールドは、プリアンブルと、透過チャンネルと、6バイトのソースデータと、制御フレームおよびSPDIF状態ビットを構成する8ビットの制御／状態ビットとを含む。次に、様々なフィールドの意味について詳細に述べる。

#### 【0019】

図6のサブフレーム構造のフィールドは下記の通りである。

・プリアンプル：プリアンプルは、ネットワークでの受信の同期をとる。プリアンプルには3つのタイプがあり、IEC-958 (SPDIF) 仕様で規定されたものと同様である。これらは、受信器が識別できる2相の符号則違反を含む。3つの一意的なプリアンプルは、左側サブフレーム、右側サブフレーム、およびサブフレームのブロックを識別する。左側プリアンプルはフレームの開始を識別し、ブロックプリアンプルはブロックの開始を識別する。ブロックプリアンプルは、左側プリアンプルが48番目になるたびにこれと置き換わる。これにより、制御フレームデータが同期するブロック構造がもたらされる。

#### 【0020】

・ソースデータバイト：ソースデータバイトは、大容量の実時間デジタルソースデータを搬送する。これらのバイトはフレキシブルに割り振ることができ、システム中のデバイスが、そのシステムにとって最も効率的な方法でソースデータバイトを使用することができる (EPA-0725520およびEPA-0725521を参照されたい)。

#### 【0021】

・制御ビット：制御ビットCF0およびCF1は、(デバイスを制御し、状態情報を送信するための) 制御メッセージを搬送する。サブフレームごとに2つのCFビットが存在し、制御フレームは192ビットの長さであり、従って、完全な制御フレームを構築するためには96個(左側48個+右側48個)のサブフレームが必要となる。制御フレームは、図7に示してある。

#### 【0022】

図5に示すように、制御フレームは、96個のサブフレームのブロックから組み立てられ、これと位置合わせされる。すなわち、新しい制御フレームの最初の2ビットは、ブロックプリアンプルを有するサブフレームからとり、後続のビット対は後続のサブフレームからとって、制御フレームを構築する。制御フレームのフィールドは、下記の通りとなる。

#### 【0023】

・調停 (arbitration) ビット：これらは、制御フレームが空いて

いるか占有されているかを示す。送信器は、これらのビットを自動的に処理する。

#### 【0024】

・送信先アドレス：これは、' 000' Hから' FFF' Hの範囲の、メッセージの宛先の12ビットアドレスである。送信側デバイスは、これを送信バッファから伝達されるメッセージ中に書き込む。いくつかのアドレスおよびアドレス範囲は特殊な意味を有し、リング位置、またはアプリケーションと関連する「デバイスアドレス」によって、ステーションのアドレス付けをすることができる。ブロードキャストおよび「グループキャスト」も提供される。

#### 【0025】

・送信元アドレス：これは、' 000' Hから' FFF' Hの範囲の、メッセージの送信側の12ビットアドレスである。受信側デバイスは、受信バッファに受信した、そのメッセージからこれを読取ることができる。

#### 【0026】

・メッセージのタイプおよび長さ：メッセージのタイプ／長さを示すために通常使用される、2つの4ビットフィールドである。メッセージのタイプは、コマンド、状態通知、および状態通知要求を含む。

#### 【0027】

・データ0から15：メッセージデータである。すべての16バイトが、常に伝達される。メッセージの長さは、通常は、16バイトのうちのどの程度が実際にそのメッセージで有効であるかを示す。送信側デバイスは、これを送信バッファから伝達されるメッセージ中に書き込む。受信側デバイスは、受信バッファに受信した、そのメッセージからこれを読取ることができる。メッセージは、通常は、演算コードおよび1つまたは複数のオペランドを含む。

#### 【0028】

・CRC：制御フレームがエラーなしで伝達されていることを検証するために使用される、16ビットの巡回冗長検査の値である。CRCは、メッセージ伝送時にインタフェースモジュールによって自動的に生成され、メッセージ受信時に自動的に検査される。

## 【0029】

・ACK/NAK：肯定応答および否定応答（それぞれ2ビット）は、メッセージの伝達が成功したことを示す。本発明者等の出願GB-A-2302243に記載されているように、別個のACKフラグおよびNAKフラグを使用することにより、信頼性の高いポイントツーポイントおよびブロードキャストのメッセージ伝達が可能となる。これらのフラグは、送信先デバイス（存在する場合）によって自動的に満たされ、送信側デバイスによって読取られる。

## 【0030】

・予約：10ビットが、将来的に規定されるために予約されている。

## 【0031】

（製品開発システムの概要）

本明細書に記載の新しい製品開発環境は、個々の製造業者またはシステムインテグレータが、製品および／または自社または他社の製品とともに動作できる完全なネットワークシステムの、開発ならびにテストを補助するために使用することができる。特に重要なものは、図7の制御フレーム機構を介して制御メッセージおよびデータを交換するためのプロトコルの適切な実装である。D2B Opticalの仕様は、メディアアクセス層からアプリケーション層のいくつかのプロトコルを（ISO/OSIネットワークモデルの用語を使用して）規定する。換言すれば、ネットワークを初期化し、制御フレームメッセージ（以下D2Bメッセージと呼ぶ）を交換するための機構が規定されるだけでなく、目的の範囲においてメッセージの特定の内容および意味も規定される。

## 【0032】

例えば、ネットワーク管理レベルでは、各ステーションがD2Bメッセージを受信するための一意的なアドレスを獲得できるように、手続きおよびメッセージのフォーマットが規定される。アプリケーションレベルでは、メッセージは、ネットワークに接続されたアンプのボリューム設定を増加させたり、または各ネットワークフレーム（図5）中のソースデータフィールドのいくつかのバイトで搬送される音声データストリームを再生するように切り替えることができるように規定される。同時に、最大限の自由が意図され、どのような種類の製品がネット

ワークに接続されても、製造業者が独自の好ましい機能を実装することができる。

### 【0033】

図8は、新しい製品開発システムの概要を与える図あり、これは特に、PC上で動作する様々なソフトウェア構成要素を含むソフトウェアエンジン800と、一または複数の異なる種類のインタフェースボード（「Optical Power Board」（OPB）804および「Advanced OPB」（AOPB）806）を載架するインタフェースボードラックシステム802とを含む。各インタフェースボードは、上述のConanチップなど、1つまたは複数のネットワークランシーバの集積回路を含む。開発システムエンジン800は、以下でさらに詳細に述べる、様々なプログラムモジュールおよびデータベースモジュール812から836を含む。システム制御モジュール832は、クライアントアプリケーションプログラム810に、アプリケーションプログラムインタフェース（API）を提供する。クライアントアプリケーション810は、この開発システムを備えた標準アプリケーションにすることも、開発中の所与の製品をエミュレートするために特別に開発することもできる。

### 【0034】

この実施形態の開発システムエンジン800は、低レベルおよび高レベルのD2B Opticalの標準速または倍速の通信、ならびにConan製品のこれらICの制御の両方とも提供する汎用サーバである。

### 【0035】

開発システムエンジンは、実際には、ActiveXなど、高度な最新のソフトウェア設計技術を使用した、プログラマブルソフトウェア構成要素の組合せである。これにより最大限のフレキシビリティがもたらされ、明確に定義されたアプリケーションプログラムインタフェース（API）を介してアクセスされる、全てのプログラムおよびアプリケーションについて高度の機能的信頼性を保持される。エンジン800は、D2B Opticalアプリケーションを書き込むことをきわめて容易にする設計システムである。これは、ランシーバICを操作し、D2B Optical制御メッセージで通信するための汎用機能を提供

する。これは、Visual Basic、Visual C++、Borland Delphiなど、Active Xをサポートする任意の環境を使用してプログラムすることができる。

#### 【0036】

このエンジンは、例えばD2B Opticalデバイスをエミュレートし、または現実のD2B Optical製品をテスト／検証するために使用することができ、また、言うまでもなくその他のプロトコルに適合させることもできる。エミュレーションだけでなく、この開発システムを形成する同じ構成要素で、ネットワークトラフィックをシミュレートおよび／または解析し、製品が標準プロトコルを遵守しているかを検証するための、共通のフレームワークを提供することができる。

#### 【0037】

ネットワーク上の現実の製品と同様に、ソフトウェア構成要素とともに使用される各インタフェースボードは、1つまたは複数のネットワークランシーバの集積回路を載架する。Advanced OPB806は、いくつかのネットワークインタフェースボードが、「デ이지チェーン」配列でラックシステム802を介して接続するため、1つのネットワーク中の複数のデバイスまたは複数の別個のネットワークを、同一PCからシミュレートまたはテストすることができる機構を提供する。以下で、ソフトウェアツールのアーキテクチャについて、いくらか詳細に述べる。理解の助けとなるように、使用中のシステムを示す2つの図を最初に与える。

#### 【0038】

図9の例では、開発システムソフトウェアは、RS-232シリアルケーブル910によって、1つまたは複数のネットワークランシーバの集積回路であるインタフェースボード920に接続された、ポータブルPC900上で動作している。このセットアップは、例えば、開発中のD2B Optical電話をシミュレートするために使用することができる。デバイスの製造業者は、現実の製品の完全な挙動モデルを例えば「C」プログラムで実装することができ、概念／設計を検証した後で、同じコードを現実の製品中の組込み型マイクロコントロー

ラに転送することができる。図1の大規模ネットワーク中の構成要素に関して言うと、製品101（ヘッドユニット）、104（CDチェンジャ）、および105（アンプ）が存在しており、開発システムは電話製品109のエミュレーション109'を提供する。

#### 【0039】

図10では、システムソフトウェアは、RS-232リンク1010を使用してOPBラックシステム1002に接続された、異なるPC1000上で動作している。OPBラックシステム自体は、ラック内で「デ이지ーチェーン」となった5個のOPBのホストとなる。このシステムは、D2B Opticalインタフェースを含む製品の、その製品が合意された互換性レベルを確実に満たすための、所定の基準セット（プロトコルおよび基本機能の挙動の検証（再生や停止など）など）に対する適合性をテストすることができる。これは、製品サイクルのどの段階でも、システムインテグレータまたは製品開発エンジニアを補助するために使用することができ、低レベルおよびアプリケーションプロトコルレベルの両方で、D2B Opticalまたはその他の適用可能な仕様に対する製品の適合性を高い確度で識別することによる。図1の参照番号を使用して、テスト中の5個のCDチェンジャ製品104-1～104-5を示す。

#### 【0040】

このようなシステムの利点は下記の通りである。

- ・テストおよびテストシーケンスの両方、ならびにテストに対する応答および不整合性の報告が予め決められているので、品質検査および妥当性検査の確度および再現性が改善される。

#### 【0041】

- ・自動テスト能力によって製品開発サイクルが短縮される。開発された製品がシステム中で正しく機能するか否か、またD2B Opticalまたは他の要求されたプロトコルが、所定のレベルまで機能するか否かを、機器が専門のテストエンジニアより正確にかつはるかに速く検査することができるからである。

#### 【0042】

- ・生産ライン上および／または生産された製品のサンプルまたは全てについて

、所定のテストサイクルに基づいて製品を検査することができる。

【0043】

・テストエンジニアが、以前に検査したデバイスが修理され、「修理」サイクルの後で、合意された標準に合わせて機能するかどうかを識別することができる。

【0044】

(開発システムのソフトウェア構造：導入)

インタフェースボードに加えて、開示のツールは、クライアントアプリケーションにサービスを提供するために統合され、相互に影響する、ソフトウェア構成要素のシステムを含み、それらのアプリケーションが、Microsoft Windowsオペレーティングシステムから、D2B Optical ネットワーク内で動作し、これを使用し、制御することができるようにする。

【0045】

以下で、このツールシステムのためのソフトウェアアーキテクチャについて述べる。システムはオブジェクト指向法で設計されるので、オブジェクトの一般的な設計、それらの目的、および全体的なシステムの他の部分との対話について議論することには意味がある。詳細な設計は、設計ツールRational Roseで実行されるが、代替ツールも同様に適している。

【0046】

(設計目標)

設計は、下記のことを念頭に置いて実行される。

・その最終目標が、D2B Opticalおよびそれと同様の技術を利用するためのソフトウェアおよび／またはファームウェアの開発を容易にすることである、ソフトウェア構成要素のファミリを生成すること。

【0047】

・低レベル制御（例えばハードウェアと直接対話する）から非常に高レベルな制御（例えば自動化されたデバイスの挙動）まで、様々なレベルの制御をユーザーに与えるようなかたちで、その範囲の構成要素を設計すること。

【0048】

- ・構成要素を階層的に設計し、より高レベルな構成要素が、その下にあるより低レベルの構成要素のサービスを利用するようにすること。これにより、構成要素およびソフトウェアを最大限に再利用することが可能となる。

#### 【0049】

- ・構成要素自体を拡張可能かつ柔軟にし、新たな要件が将来発生した場合に、それらの新しい変更形態を容易に作成できるようにすること。

#### 【0050】

- ・構成要素のエンドユーザをハードウェアから抽象化すること。これにより、ユーザは、その基礎となるハードウェアについて心配することなく、またいつそのソフトウェアを書き込むかを考慮する必要なく、ソフトウェアを書き込むことが可能となる。

#### 【0051】

- ・構成要素を利用するための簡単かつ分かりやすいインタフェースをユーザに提供すること。

#### 【0052】

- ・その他のエミュレータ、コンプライアンスチェッカ、およびユーザ定義（専有の）エミュレータに、その基礎となる信頼できるアーキテクチャを提供すること。

#### 【0053】

- ・以下で説明する、自動応答システムを提供すること。

#### 【0054】

##### （クラス設計）

以下のセクションでは、この設計のオブジェクトについて論じる。この考察は一般的なものであるが、必要な箇所は具体的にしている。当業者には周知のオブジェクト指向アーキテクチャの用語では、「オブジェクト」の様々な「クラス」が定義される。各クラスの1つまたは複数の「インスタンス」は、明確に定義された方法で別個の「オブジェクト」として互いに作用し、対話することができる。図8は、階層的なクラスを示しており、「低レベル」構成要素は最下部に、より高レベルな構成要素は最上部にある。これらの構成要素または「オブジェクト

」は、メッセージの交換によって対話することができる。矢印は、システム内の通信の主な流れを説明している。

#### 【0055】

##### (OPBドライバ812)

OPBドライバ812のオブジェクトは、実際のネットワークランシーバのハードウェアとの最低レベルのインタフェースを提供する。各オブジェクトは、特定のインタフェースボードの型、例えば並列、直列構成、A-OPB等とのAPI（アプリケーションプログラムインタフェース）を提供する。OPBドライバクラスは、低レベル通信および制御機構の知識をカプセル化し、それらの機能にアクセスするためのオブジェクト指向インタフェースを提供する。さらに、各クラスが、それがカプセル化したドライバ中のエクスポート関数それぞれについてメンバ関数を提供すれば、既存のアプリケーションを将来的にクラスを使用するために改変することができる。

#### 【0056】

任意の新しいインタフェースボード（例えばUSBまたはPCIインタフェースを備えた）が開発されると、その新しいインタフェースボードを実際に制御するドライバについて、新しいOPBドライバのクラスも再度設計しなければならない。

#### 【0057】

##### (OPB814)

OPBクラスは、エンドユーザのアプリケーションおよび開発システム自体が、Optical Power Boardファミリーの中のハードウェアボードを制御するために使用するクラスである。従って、ユーザが利用できる機能は、OPBクラスがそのAPIの中で表明する(expose)機能に制限される。OPBオブジェクトはそれぞれ、関連する型のOPBドライバオブジェクトに接続される。OPBオブジェクトは、ドライバオブジェクトを使用して、OPBファームウェアとの実際の通信を実行する。OPBオブジェクトは、ディレクタ構成要素（以下参照）によって、OPBドライバへのアクセスを与えられる。ディレクタ構成要素は、全てのOPBオブジェクトを作成し、最終的に破棄するオ

プロジェクトである。

#### 【0058】

OPBクラスの主な目的は、OPBハードウェアとエンドユーザの間に抽象化の階層を提供することである。これにより、ユーザは、汎用ソフトウェア機構を使用して、任意の構成におけるOPBの任意の型を制御することが可能となる。その結果として、ユーザソフトウェアの複雑さが軽減される。この抽象化は、全てのOPBクラスが関数の共通のインタフェースを実装することを保証することによって実現される。この場合、ユーザは、プログラムでこのインタフェースを使用して、OPBオブジェクトを制御する。ユーザは、OPBを制御するためにOPBの型またはその構成を知っている必要はない。

#### 【0059】

抽象化のその他の大きな利点は、インタフェースボード（例えばUSB、PC（PCMCIA）、PCI、またはISAバスを介して）と通話するために新しい通信機構が開発する場合に、この新しい構成中のOPBと通話するために、OPBクラスのソフトウェアをいかなる形でも改変する必要がないことである。それよりむしろ、どのようにしてこの新しい機構を使用して通信するかについての必要な知識をカプセル化する、新しいOPBドライバクラスを単に使用することになる。

#### 【0060】

OPBクラスは、現実の物理的インタフェースボードをソフトウェアの形で表そうと試みる。物理的OPBは、ネットワークランシーバや音声コーデックIC、マイクロコントローラファームウェアなどの構成要素を含む。これらは、D2B Opticalの開発者が、操作および制御したいと望む構成要素である。従って、OPBクラスは、対応するハードウェア構成要素を表す補助的なソフトウェア構成要素も含む。これらのソフトウェア構成要素は、本質的にクラスである。このシステムでは、このような4つのクラスを定義する（以下でより詳細に述べる）。

- ・ネットワークランシーバ816
- ・コーデック818

- ・ネットワーク管理820
- ・インタフェースファームウェア822

#### 【0061】

OPBオブジェクト814は、補助的な構成要素オブジェクトを作成し、これは、インタフェースボードの型およびどのハードウェアが実際のOPB上にあるかに依存している。これは、OPBオブジェクトのユーザに、そのAPI中の機能を介したこれら他の構成要素オブジェクトを与える。ユーザは、適当な構成要素オブジェクトを使用して、物理的インタフェースボード上のハードウェア構成要素の機能にアクセスすることができる。

#### 【0062】

一例として、インタフェースボードにおけるOPBの型は、音声コーデックCS4225ICおよびネットワークランシーバOCC8001ICの上に有る。OPBオブジェクト814が作成されたときに、ネットワークランシーバオブジェクト816およびコーデックオブジェクト818を作成する。OPBクラスは、これらのオブジェクトへのアクセスを可能にするために、そのAPI中の関数GetComponent(“Network Tranceiver”, 0)、およびGetComponent(“Codec”, 0)を表明する。ネットワークランシーバおよびコーデックのクラスはAPIを表明し、ユーザが物理的インタフェースボード上の対応するハードウェアを制御することができるようにする。インデックス(ここでは‘0’)は、ボード上の構成要素の数を示す。これにより、1つのボード上で複数の同一の構成要素を許容する(例えば2つのランシーバICを備え、その第2は、GetComponent(“Network Tranceiver”, 1)でアクセスされる)。

#### 【0063】

この手法は以下のいくつかの利点を与える。

- ・モジュラ分解。OPBクラスの機能性をいくつかの別個のソフトウェアモジュールに分割する。従って、OPBの複雑さをより単純な構成部分に分割することができる。これにより、ソフトウェアがより単純になり、保守もより容易になる。

## 【0064】

・構成部品の再利用。構成要素が別個のモジュールであるので、その他のソフトウェアモジュールの間でこれらを容易に共用することができる。例えば、ネットワークランシーバおよびコーデックのクラスは、同等のICがボードの両方の型に存在することから、OPBおよびAOPBのクラス双方によって使用される。構成要素のためのコードは集中化され、構成要素の変更は一箇所で行うだけでよい。

## 【0065】

・拡張性。上記の点の結果として、ハードウェアの新しい要素が、OPB（例えばDSP IC）に追加された場合には、単にそのための新しい構成要素クラスを開発し、これをOPBクラスに「差し込む」だけとなる。このプロセスは、OPBクラス自体をほとんど変更せずに実行することができる。

## 【0066】

OPBによって作成された構成要素オブジェクトはそれぞれ、最終的にインタフェースボードファームウェアと通話して、所望の動作をハードウェア上で実行しなければならない。このファームウェアとの全ての通信は、OPBドライバオブジェクトを介して行われなければならないので、各構成要素が関連するOPBドライバオブジェクト812にアクセスすることが必要と考えられる。しかしながら、これは、OPBクラスを第1に有するという利点を損なうことになる（ファームウェアとの通信機構を抽象化するために）。各構成要素は、任意の構成であらゆる型のOPBと通話するために、固有のコードを実施しなければならない。

## 【0067】

この問題を軽減するために、OPBの構成要素オブジェクト816、818の一方から実際のインタフェースボードへの全ての通信は、OPBオブジェクト814自体を介して行われる。これはつまり、構成要素オブジェクトが、OPBの特定の知識および通信機構をそれらのコードに埋め込む必要がないということである（これがOPBオブジェクト中にあるため）。しかしながら、OPBクラスが、ファームウェアと通話するためには、これらの構成要素オブジェクトについ

て厳密に定義された（ただし多様な）インタフェースを表明しなければならないことを意味する。構成要素オブジェクトは、OPBクラスのどの型にそれらが接続されるかについて仮定を立てることができない。様々な型のOPBオブジェクトに接続される可能性があるからである（上述した構成要素の再利用の点を参照のこと）。しかし、構成要素オブジェクトは、それらが接続される任意の型のOPBクラスとどのようにして通話するかを知らなければならない。これはつまり、全てのOPBクラスのオブジェクトが、それらの構成要素について定義されたインタフェースを提供しなければならないということである。

#### 【0068】

従って、このシステム中のOPBクラスは、インタフェースボード上の特定の機能にアクセスするために、他の構成要素のいくつかのインタフェースを表明する。OPBクラスは、実際には、4つの異なるインタフェースを実装する。

#### 【0069】

- ・クライアントインタフェース。これは、エンドユーザのアプリケーションに対して表明される主要インタフェースであり、これを介して、それらのアプリケーションはOPBの全ての機能（より正確にはOPBがエンドユーザにアクセスを許す全ての機能）にアクセスすることができる。

#### 【0070】

- ・レジスタインタフェース。このインタフェースは特に、ネットワークトランシーバおよびコーデックオブジェクトが、ネットワークトランシーバIC中でレジスタの読取り／書込みを実行するために使用する。

#### 【0071】

- ・トランシーバサポートインタフェース。このインタフェースは、ネットワークトランシーバオブジェクト816が、D2Bメッセージを送信し、OPBオブジェクトから取出し、そのD2Bアドレスを設定するために独占的に使用する。

#### 【0072】

- ・制御インタフェース。これは、ディレクタ824が、OPBオブジェクトを初期化し、操作するために独占的に使用する。

#### 【0073】

OPBの何らかの特定の機能にアクセスするためには、クライアント（ユーザ）は、それを提供する適当な構成要素オブジェクトを経由せざるを得ず、クライアントインタフェースを経由することはできない。換言すれば、OPB上のトランシーバICへのレジスタ書込みを実行するためには、ユーザはネットワークトランシーバオブジェクト816を使用しなければならない。主要なOPBクライアントインタフェースは、トランシーバレジスタへの書込みを直接行うための機能を表明しない。この考えを実施することにより、特定の一意的な動作を実行するための方法がただ1つとなることが保証される。これは一般にソフトウェアを単純化し、保守をより容易にする。

#### 【0074】

この開発システムのもう1つの重要な側面は、それが、インタフェースボード上およびより広域ネットワーク中で発生したイベントを、エンドユーザのアプリケーションに通告することである。これらのイベントは、OPBファームウェアによってOPBドライバオブジェクト812に渡され、次いでこれがそれらのイベントをOPBオブジェクト814に渡す。OPBオブジェクトはこれらのメッセージを復号し、開発システムの残りの部分で認識されるフォーマットに変換する。OPBクラスだけが、OPBドライバから渡されたメッセージの意味を理解する。従って、特定の型のインタフェースボードのOPBドライバが、これらのメッセージについて同一のフォーマットを使用することが重要である。これは、OPBドライバクラスからOPBクラスへの抽象化を維持するために重要である。

#### 【0075】

OPBはメッセージを復号し、これをネットワークトランシーバオブジェクト816に渡し、そこからメッセージは最終的にエンドユーザのアプリケーションに伝達されることになる。

#### 【0076】

（ネットワークトランシーバ816）

ネットワークトランシーバクラスは、インタフェースボード上の実際のネットワークトランシーバICを制御し、ネットワークトランシーバに関する情報を取

出すためのAPIをもたらす。例えば、ConantランシーバIC、型番OC8001の知識をカプセル化するネットワークランシーバクラスがある。

#### 【0077】

ネットワークランシーバクラスの主な目的は、エンドユーザとOPB上のランシーバICとの間に、抽象化した階層を提供することである。この考えは、ユーザが、任意タイプのネットワークランシーバの機能にアクセスするために汎用ソフトウェアを使用できるようにする点で、この前のセクションで詳述したOPBクラスの抽象化の考えとほぼ全く同じである。その結果として、ユーザは、単一のソフトウェアを書き込めば、任意のネットワークランシーバクラスの機能にアクセスすることができる。この抽象化は、全てのネットワークランシーバクラスが、諸機能の共通のインタフェースを実装することを保証することによって実現される。この場合ユーザは、ネットワークランシーバオブジェクトを制御するためのプログラミングによってこのインタフェースを使用する。ユーザは、ネットワークランシーバオブジェクトを使用するために、そのいかなる特定の知識も有する必要がない。

#### 【0078】

ネットワークランシーバクラスは、4つの主要な機能領域へのアクセスを提供する。

- ・レジスタの読取りおよび書込み。
- ・レジスタについての情報（例えばそれらの名前など）の取出し。
- ・ネットワークランシーバについての情報（例えばその型や、多くのレジスタをどのようにして有するかなど）の取出し。
- ・D2Bメッセージの送信および取出し。

#### 【0079】

最初の3つの機能領域は、全てのネットワークランシーバクラスに共通の、主要な汎用インタフェースに含まれる。これは、クライアントアプリケーション810に表明されたAPIである。他の1つのAPIが、D2Bメッセージの送信/取出しのために設けられる。このAPIは、クライアントが直接使用することができず、開発システムのその他の構成要素（すなわちメッセージ制御構成要

素)によって内部で使用されるだけである。

#### 【0080】

その理由は、やはり抽象化と関係がある。現在、トランシーバICは、D2B Opticalの制御フレームフォーマットを使用している。将来、異なるプラットフォームに適合するように制御フレームフォーマットが変更された場合には、そのD2Bメッセージの送信/取出しのためのAPIは、その他のネットワークトランシーバオブジェクトと異なることになる。これは、APIを抽象化することができないことを意味する(メッセージを送信する機能は、ネットワークトランシーバごとに異なるので)。その代わりにD2Bメッセージの概念を抽象化することができ、ネットワークトランシーバクラスはそれらを送信するための共通のAPIを実装することになる。しかし、そのためには、ネットワークトランシーバクラスがD2Bメッセージの概念を理解する必要があり、これはネットワークトランシーバクラスのさらに高いレベルの概念であると判断されている。さらに、D2Bメッセージを送信するために余分な制御コードが必要となり、これをあらゆるユーザが書き込むことは期待できない。従って、D2B OpticalメッセージへのアクセスのAPIは、エンドユーザに与えられない。その代わりに、アプリケーションは、D2Bメッセージの送受信のために、システム制御構成要素のAPIを経由しなければならない。

#### 【0081】

ネットワークトランシーバオブジェクト816は、開発システム環境中のOPBオブジェクトによって作成され、破壊され、所有される。ネットワークトランシーバオブジェクトは、OPBオブジェクトのレジスタインタフェースおよびトランシーバサポートインタフェースとの接続を維持し、それを介してOPBファームウェア822にアクセスする。クライアント810は、書き込まれたレジスタ、およびそれらに書き込まれた値を通知される。OPBオブジェクト814は、イベントをネットワークトランシーバオブジェクトに渡すこともできる。これを行うために、ネットワークトランシーバクラスは、OPBが使用するための共通の汎用APIを提供する。このAPIは(抽象化のために)汎用でなければならず、従ってOPBオブジェクトはそれらのトランシーバを透過的に取り扱うこと

ができる（すなわち、それらは同じ機構／コードを使用して、任意の型のトランシーバにイベントを通知することができる）。

#### 【0082】

OPBオブジェクト814からのイベントは、最終的にクライアントアプリケーション810に達しなければならないので、ネットワークトランシーバオブジェクト816は、何らかの方法でメッセージをアプリケーションまで伝達させなければならない。ネットワークトランシーバオブジェクトは、メッセージ制御オブジェクトを介してこれを行う。メッセージ制御オブジェクトは、OPBハードウェアを表す構成要素（OPB、トランシーバ、コーデックの諸構成要素）と、「システムインスタンス」構成要素（メッセージ制御、自動応答システム（ARS）、デバイスの諸構成要素）との間の「ブリッジ」である。「システムインスタンス」構成要素は、この開発システムで使用する各アプリケーションに一意的である。各アプリケーションは、メッセージ制御、ARS、デバイス、およびシステム制御の諸構成要素の、それ自体一意的なインスタンスを有することになる。これは、開発システムのアプリケーション間で共用される「ハードウェア構成要素」（OPB、トランシーバ、コーデックなど）とは対照的である。これらハードウェアオブジェクトの各々のインスタンスの1つが、現実のハードウェアに対応してそれぞれ存在する。

#### 【0083】

従って、メッセージ制御オブジェクト826は、ハードウェア構成要素と開発システムの構成要素間のブリッジを提供する。ネットワークトランシーバオブジェクトは、メッセージ制御オブジェクトに接続し、および切断する能力を有する。これを行う機能は、ネットワークトランシーバオブジェクト上の、D2Bメッセージ機能と同じインタフェース中にある。「接続」を確立するプロセスでは、ネットワークトランシーバオブジェクトは、受信したOPBイベントおよびD2Bメッセージのメッセージ制御オブジェクトに、トランシーバが通知するために使用する、メッセージ制御オブジェクト上のインタフェースの参照を獲得する。接続が確立された後で、クライアントアプリケーションは、D2Bメッセージをインタフェースボードハードウェアへ送信し、および受信することができる。接

続を行うためのネットワークランシーバクラス上のAPIは、やはり汎用でなければならず、従って、メッセージ制御オブジェクトは、それが接続しようとするランシーバの型について気にする必要なく、あるいはそれを知る必要もなく、任意の型のネットワークランシーバオブジェクトに接続することができる。

#### 【0084】

ネットワークランシーバクラスの1つの最終的なサービスは、開発システムのネットワークランシーバオブジェクトを使用して、レジスタ書込みがランシーバIC上で実行されたときに、クライアントアプリケーションに通知することである。これが起こると、ネットワークランシーバオブジェクトがメッセージを生成し、これをメッセージ制御構成要素826に渡し、最終的にクライアントアプリケーションに渡す。このメッセージは、書き込まれたレジスタ、およびそれに書き込まれた実際の値を示す情報を含む。

#### 【0085】

(コーデック818)

ネットワークで記述される型は、制御メッセージおよびデータメッセージに加えてソースデータストリームも生成するので、コーデッククラスは、インタフェースボード上の音声または画像コーデックICを制御し、コーデックに関する情報を取出すためのAPIを提供する。例えば、Codec CS-4225 ICの知識をカプセル化するコーデッククラスがある。

#### 【0086】

ネットワークランシーバクラスの場合と同様に、コーデッククラスの主な目的は、開発システムのアプリケーションとOPB上に存在することがある様々なタイプのコーデックICとの間に、抽象化の階層を提供することである。コーデックは、以下の機能領域を有する。

- ・コーデックのプロパティの設定／取出し。
- ・レジスタの読取りおよび書込み。
- ・レジスタについての情報（名前など）の取出し。
- ・コーデックについての情報（そのタイプや、どの位のレジスタを有するかなど）の取出し。

## 【0087】

この実施形態のコーデックオブジェクト818は4つのプロパティを有する。

- ・ミュート (m u t e)
- ・ボリューム
- ・バランス
- ・フェード

これらのプロパティは、コーデックIC自体の属性ではなく、開発システムが与える概念である。これは、いくつかのコーデックICレジスタ（すなわち出力減衰レジスタ）の意味を、ユーザに対して意味をなすレベルまで抽象化するものである。換言すれば、コーデックICの出力減衰レジスタが全てゼロであるならば、これはエンドユーザに対して意味をなさない。そうではなく、コーデックICのボリュームが100%であると言え、これはユーザに対して関連した意味をなす。このように、ボリュームとは減衰レジスタの抽象表現であり、ユーザに対して意味をなす。これらのプロパティは、コーデックオブジェクトの主要なクライアントAPIを介して設定し、取出すことができる。

## 【0088】

コーデック構成要素は、常に、そのOPBオブジェクトに生成され、ネットワークランシーバオブジェクトときわめてよく似た方法で、OPBオブジェクトによって接続される。コーデックオブジェクト818は、OPBオブジェクトのレジスタインタフェースを使用してOPBファームウェアと通話し、レジスタ書き込みを実行する。コーデックオブジェクト818を使用してコーデックICに対してレジスタ書き込みが行われると必ず、コーデックオブジェクトは、書き込まれたレジスタ、およびそれらに書き込まれた値をクライアントアプリケーションに通知する。コーデックオブジェクトは、OPBオブジェクトのレジスタインタフェース中の、別の機能呼び出すことによってこれを行う。コーデックオブジェクトは、OPBオブジェクトによってイベントを通告されることはない。

## 【0089】

(ディレクタ824)

ディレクタオブジェクト824は、本質的に、インタフェースボードマネージ

構成要素である。1つだけのディレクタのインスタンスが、いつでも実行／存在することになる点で一意的である。これにより、集中化されたインタフェースボードの管理がもたらされる。ディレクタは、実行中の任意のOPBオブジェクトにクライアントがアクセスできるようにする機能を含む、単一のAPIを表明する。これを行うことができるようにするために、ディレクタは、その環境中に存在する全てのOPBオブジェクトについて知っている。ディレクタは、全てのOPBオブジェクトを実際に作成する（かつ最終的に破壊する）のがディレクタオブジェクト自体であるという事実の効能により、この知識を有する。これはまた、ディレクタがOPBクラスの全ての型、およびそれらを作成する方法についての特定の知識を有することも意味する。

#### 【0090】

ディレクタが作成されたときに最初に行うことは、どのインタフェースボードがPCに取り付けられたかを決定することである。ディレクタは、可能なあらゆる型のOPBドライバオブジェクトのインスタンスを作成し、それらを使用して実際のOPBを検出することによってこれを行う。Optical Power Boardが検出された場合には、ディレクタは適当なOPBオブジェクトを作成し、そのOPBオブジェクトをOPBドライバオブジェクト812にリンクする。その後、OPBオブジェクトは、OPBドライバを介して自由にOPBと通信することができる。ディレクタは、それらを検出することができなくなるまで、指定された各型のドライバおよびOPBオブジェクトを作成し続ける。

#### 【0091】

例えば、ディレクタが作成されたときに、並列OPBドライバオブジェクトを作成する。次いで、ディレクタはそのオブジェクト中のOPB検出機能呼び出す。この機能は、PCに並列に取り付けられたOPBが存在しないことを意味する「False」を返し、従ってディレクタは並列OPBドライバオブジェクトを破棄する。次いで、直列OPBドライバオブジェクトを作成し、そのOPB検出機能呼び出す。この機能は、シリアルRS-232で接続されたOptical Power BoardをPCが有することを意味する「true」を返す。従って、ディレクタはOPBオブジェクト814のインスタンスを生成し（

instantiates)、これに直列OPBドライバオブジェクトへの参照を渡す。ディレクタは、複数存在することはないという明示的な知識を有するので、それ以上の直列OPBの検出を試みようとはしない。最後に、ディレクタは、Advanced OPB (AOPB) ドライバオブジェクトを作成し、これを用いてAOPBを検出する。AOPBの構成要素オブジェクトが作成され、AOPBドライバオブジェクトに結合される。Advanced OPBは、図10のセットアップで説明したデージーチェーン機構を提供する。ディレクタは、その他のAOPB (例を図10に示す) が存在することがあることを知っている。別のAOPBドライバ構成要素を作成し、別のAOPBの検出を試行する。ディレクタは、PCに取り付けられた全てのインタフェースボードを検出するまで、引き続きこれを行う。

#### 【0092】

ディレクタは、インタフェースボードの検出に成功した、全てのOPBオブジェクトへの参照を記憶する。そして、ディレクタは、任意の要求側クライアントにこれらの参照を提供することができる。

#### 【0093】

ディレクタがオプションとして有するよう構想された唯一のその他の機能領域は、OPBオブジェクトと同様にデバイスオブジェクト830等を指図することである。これにより、アプリケーションが、実行中のその他の開発システムを「スパイ」することが可能となる(デバイスのセクションで述べる)。これを行うために、ディレクタオブジェクトは、通常、デバイスオブジェクトがディレクタにそれら自体を登録および登録解除することができるようにする機能と、登録されたデバイスオブジェクトへの参照をアプリケーションが取出すための機能とを備えた、もう1つのAPIを表明することになる。

#### 【0094】

(メッセージ制御826)

メッセージ制御構成要素クラスは5つの主要な機能カテゴリを提供する。

- ・D2Bメッセージのキューイングおよび送信。
- ・D2B制御フレームの受信および変換。

- ・ O P B イベントの受信および伝達。
- ・ 長い D 2 B   O p t i c a l メッセージの管理。
- ・ トランシーバオブジェクトおよび A R S オブジェクトへの接続の生成および破棄。

#### 【0095】

メッセージ制御クラスの主な機能は、D 2 B メッセージのキューイングおよびクライアントアプリケーションへの送信サービスを提供することである。メッセージを送信できるようにするためには、最初に、メッセージ制御オブジェクトをネットワークトランシーバオブジェクト 8 1 6 に接続しなければならない（かつそれにより間接的に物理的インタフェースボード 8 0 4 / 8 0 6 などに）。メッセージ制御クラスは、それをネットワークトランシーバオブジェクトに接続する機能およびそこから切断する機能を備えたインタフェースを表明する。メッセージ制御オブジェクトには、ネットワークトランシーバオブジェクトのインタフェースへの参照が与えられ、これを介して、メッセージ制御オブジェクトは接続されているネットワークトランシーバオブジェクトの型を決定する。そして、メッセージ制御は、ネットワークトランシーバオブジェクトに、そのインタフェースのうちの 1 つへの参照を与える。このインタフェースは、そのネットワークトランシーバオブジェクトが後に、イベントおよび D 2 B メッセージをメッセージ制御オブジェクトに渡すために使用するインタフェースである。メッセージ制御オブジェクトは、いくつかの異なるこのような「通告」インタフェースを表明することができる。これにより、下記のように、将来の拡張性および柔軟性をもたらすことができる。

#### 【0096】

新しい開発システムは、拡張可能かつ「将来に備えた（f u t u r e   p r o o f）」ものとなるように意図されている。メッセージ制御クラスの意義の 1 つは、D 2 B 制御フレーム構造（図 7）が将来変化することができること（おそらく例えば新しい型のトランシーバ I C のために）、または同じプロトコルが異なる型のネットワーク上で完全に実装されることになることである。従って、メッセージ制御クラスは、様々な制御フレームフォーマットを使用していることが

ある、様々なネットワークランシーバオブジェクトと対話することができなければならない。しかし、開発システムの構成要素およびクライアントアプリケーションは、制御フレームではなくD2Bアプリケーションプロトコルのメッセージの概念を扱う。D2Bアプリケーションプロトコルメッセージは、メッセージの現実のセマンティクス（意味）を含むが、制御フレーム構造は、ランシーバハードウェアによってアプリケーションプロトコルメッセージに付与された単なる通信フォーマットである。開発システムのエンドユーザは、D2Bメッセージを送信することに関心を持つが、D2B光ネットワークを介してこれらのメッセージを送信するための実際のフォーマットを覚えていない。D2Bメッセージは、それを通信する際に基礎となった物理フォーマットからメッセージの意味を分離（abstract）するものとして考えることができる。

#### 【0097】

従って、クライアントアプリケーションおよび開発システムの構成要素はD2Bメッセージの概念を使用するが、同じフレームフォーマットがアプリケーション中でその他の型のネットワークまたはその他のプロトコルに適用されることはない。「ハードウェア」構成要素（ランシーバおよびOPBクラス）は、制御フレームを扱う。開発システムのフレームワーク中のどこかで、D2Bメッセージを制御フレームに、またその逆に翻訳しなければならず、これはメッセージ制御クラスの仕事である。メッセージ制御オブジェクト826は、全ての制御フレームフォーマットを知っている。メッセージ制御オブジェクト826は、最初にネットワークランシーバオブジェクト816に接続されたときに、ランシーバの型を決定し、それによりその制御フレームフォーマットを決定する。後にD2Bメッセージが送信されたときに、メッセージ制御はこのD2Bメッセージを、ランシーバが期待する適切な制御フレーム構造に変換することができる。様々なネットワークランシーバクラスが、様々な制御フレームの型を送信するための様々なAPIを表明することになる（従ってこのAPIは抽象化することができない）。これが、エンドユーザのアプリケーションが、この実施形態で、ネットワークランシーバクラスのAPIへ送信するメッセージへのアクセスを与えられない理由である。

## 【0098】

制御フレームが受信されると、ネットワークランシーバオブジェクト816は、これをメッセージ制御オブジェクト826に渡す。様々なタイプのランシーバが様々な制御フレームを使用するので、メッセージ制御オブジェクトは、存在する制御フレームの型それぞれについてAPIを提供する。メッセージ制御がネットワークランシーバオブジェクトに接続されたときに、適当なメッセージ制御の通告インタフェースへの参照をこのランシーバに与え、ランシーバは、これを介してその特定の制御フレームの型をメッセージ制御に渡すことができる。この最終的な結果として、メッセージ制御構成要素は、任意タイプのランシーバICを介してD2Bメッセージを送受信することができる。同時に、開発システムの構成要素およびエンドユーザのアプリケーションは、D2Bメッセージの考えを引き続き使用することができ、将来新しい型のランシーバを利用するためにソフトウェアを改変する必要はない。

## 【0099】

アプリケーション810がD2Bメッセージを送信すると、メッセージ制御オブジェクト826は、このメッセージを直ちに伝送しようとはせずに、バッファに入れる。メッセージ制御オブジェクトが接続されたネットワークランシーバオブジェクト816は、定期的にメッセージ制御オブジェクトに信号を送り、それが進行することができ、望むならD2Bメッセージの送信を開始することができることを示す。そして、メッセージ制御オブジェクト826は、そのバッファから自由にD2Bメッセージを取出し、送信する。これは次いで、送信結果を取出し、クライアントアプリケーションに通知する。メッセージをバッファ中に置くことで、開発システムは、追加の通信管理を実行し（例えば以下の長メッセージを参照されたい）、アプリケーションがそれ自体でバッファリングを実装する実施する必要を防ぐ。

## 【0100】

ネットワークランシーバオブジェクト816は、着信する制御フレームを受信すると、これをメッセージ制御オブジェクト826に渡す。メッセージ制御オブジェクト826は、制御フレームをD2Bメッセージに変換し、それをクライ

アントアプリケーション810に渡す。

【0101】

従って、メッセージ制御オブジェクト826は、以下の3つのことを開発システムのクライアントアプリケーションに通知する役割を担う。

- ・D2B Opticalメッセージの受信。
- ・送信されたD2B Opticalメッセージの送信結果。
- ・基礎となる開発システムのハードウェア構成要素から受信したイベント。

【0102】

これを達成するために、メッセージ制御オブジェクトは、ネットワークトランシーバをメッセージ制御オブジェクトに接続したのと同様に、ARSオブジェクト(828、後述)に接続することができる。これを行う機能も、ネットワークトランシーバオブジェクトがメッセージ制御オブジェクトに接続するために使用したのと同じAPI中にある。

【0103】

従って、メッセージ制御オブジェクト826は、以下2つのAPIを有する。

- ・クライアント/制御インタフェース。
  - ・メッセージ制御をネットワークトランシーバオブジェクトに接続。
  - ・メッセージ制御をARSオブジェクトに接続。
  - ・D2Bメッセージを送信。

【0104】

- ・接続されたトランシーバが使用する通告インタフェース。
  - ・受信した制御フレームをメッセージ制御オブジェクトに渡すこと。
  - ・ハードウェア構成要素メッセージをメッセージ制御に渡すこと。
  - ・それが自由にD2Bメッセージを送信できることをメッセージ制御に通知すること。

【0105】

この例でメッセージ制御クラスが提供する最後のサービスは、長いD2Bメッセージに関する。長いD2Bメッセージは、単一の制御フレームで伝送することができず、従って長いD2Bメッセージを複数の制御フレームにパッケージ化し

、伝送するために、余分な管理コードが必要となる。同様の管理コードが、長いD2Bメッセージの中に制御フレームを蓄積しパッケージ化するために、長いメッセージを受信する側でも必要となる。さらに、制御フレームの送受信において、いくつかのプロトコル（例えばフロー制御を補助するためのプロトコル）が必要となることもある。

#### 【0106】

全ての長メッセージの論理は、長メッセージを扱おうとするあらゆる開発システムのアプリケーションによって実装されなければならないので、その代わりにメッセージ制御クラスがこれを実装する。メッセージ制御クラスは、long message protocol (LMP) クラスのオブジェクトを使用して、これを内部で行う。このオブジェクトは、長メッセージの送受信に必要な全ての知識を含む。このクラスは、将来開発される任意の新しい長メッセージプロトコルを使用するために更新することができる。現在の実装は、制御フレームをD2Bメッセージに（またその逆に）パッケージ化／パッケージ解除するための論理を含み、またフロー制御プロトコル論理を含むように適合させることもできる。最終的な結果として、常に、ユーザは長メッセージプロトコルから防護され、比較的容易にかつ余分なコーディングを行わずに、それらが長メッセージを送受信することができるようにする。特定のlong message protocolを使用する、または長メッセージの処理を完全に止める命令を、メッセージ制御オブジェクトにするための手段が存在することもできる。

#### 【0107】

メッセージ制御オブジェクトは、長メッセージを受信したときには、さらに処理するためにこれをLMPオブジェクトに渡す。LMPオブジェクトは、必要な任意の動作を実行し、制御フレームから完全なメッセージを構築する。メッセージが完成すると、通常の方法によってARSオブジェクト828に渡される。長メッセージの送信中には、LMPオブジェクトは、D2Bメッセージオブジェクト自体といっしょに、このメッセージを管理し、メッセージを送信するための制御フレームにパッケージ化する。全ての制御フレームが送信された後で、クライアントアプリケーションにその送信結果が通知される。

## 【0108】

## (ARS (自動応答システム) 828)

ARSクラスは、開発システムのクライアントアプリケーションに知的自動応答を付加するための基礎を提供する。このクラスは以下の機能領域を提供する。

- ・システムおよびデバイスの仕様。
- ・D2Bメッセージの妥当性検査。
- ・D2Bメッセージの翻訳。
- ・自動デバイスの挙動／応答。

## 【0109】

本開発システムの背景にある動機の一つは、D2B Opticalの製品の各設計者／製造業者が、最初に独自のD2Bネットワークシステムを開発するという事実である。このシステムは、D2B Opticalのプロトコルの指定されたサブセットを使用して通信する、指定されたD2B Opticalデバイスを含むことになる。D2Bシステムインテグレータは、システム中のデバイスがプロトコルおよびシステム仕様に適合すること（例えば、それらがそのシステムの実際の構成では無効なD2Bメッセージを送信しないこと）をテストできることを望む。D2Bシステムインテグレータはまた、デバイスエミュレータ、および自分のD2B OpticalシステムまたはD2B Opticalデバイス／製品のテストを自動化するためのテストアプリケーションを構築することも望むことがある。

## 【0110】

開発システムでこの考えをサポートするための中核となるのは、システム記述ファイル(SDF) 838中でカプセル化されたプロトコルデータベースの概念である。SDFは、以下のことを行う。

- ・特定のD2B Opticalバージョンについて全てのアプリケーションプロトコルを含む。
- ・その特定のシステムで有効なプロトコルのサブセットを指定する。
- ・そのシステムで有効なデバイスを指定する。
- ・デバイス中の有効なサブデバイス、およびそれらサブデバイスのソースデータ

の型を指定する。

#### 【0111】

このように、SDFは、デバイス（CDチェンジャや電話など）が実際にその中に存在する、任意の特定のD2B Opticalシステムの仕様をカプセル化するために作成することができる。ARSに、それが表明するAPI中の機能を介して、特定のSDFを使用するよう命令することができる。そして、ARSはSDF中の知識を利用することができ、またその他の開発システムの構成要素がその同じ情報にアクセスできるようにするための、もう1つの別個のインタフェースも表明する。

#### 【0112】

図8から分かるように、クライアントアプリケーション810との全てのD2Bメッセージ通信は、ARSオブジェクト828を通過しなければならない。従って、ARSオブジェクトは、送信または受信されるあらゆるメッセージを検査することができ、プロトコルおよび（この特定のシステムでは）SDF中に記述されたシステムの仕様にそのメッセージが適合するかどうかをテストする。クライアントアプリケーションが無効メッセージを送信しようとした場合には、ARSは、このメッセージを拒絶する（送信しない）ことも、単にアプリケーションに警告することもできる。無効メッセージが受信された場合には、ARSはそれが無効であるとクライアントアプリケーションに警告することができる。無効メッセージに応答したARSオブジェクト828の挙動は、ARSのAPI中の機能を介して構成することができる。従って、SDFを使用することにより、ARSは、自動化されたD2Bメッセージの妥当性検査サービスをクライアントアプリケーション810に提供することができ、クライアントはいかなる作業も実行する必要がない。

#### 【0113】

ARSのもう1つの特徴は、メッセージの記号翻訳（symbolic translation）である。各D2Bアプリケーションプロトコルメッセージは、例えば、演算コードおよびいくつかのオペランドとして指定される。これは複雑なために、エンドユーザにとっては非常に分かりにくい表現であり、また数

字列の意味を覚えることは困難である。これを多少とも解消するために、新しい開発システムは、D2Bメッセージを人が読めるテキスト表現に、またその逆に翻訳することができる。このサービスは、ARSオブジェクト828によって提供される。従って、クライアントアプリケーション810は、送信すべきD2B

Opticalメッセージをテキストで指定することができ、ARSオブジェクトは、このテキストを対応するD2B Opticalメッセージに変換することになる。同様に、ARSオブジェクトが受信したD2Bメッセージは、クライアントアプリケーションに伝達される前にテキストに変換することができる。ARS中のテキスト翻訳は、そのAPI中の機能によってオンまたはオフにすることができる。

#### 【0114】

ARSオブジェクト828は、メッセージ制御オブジェクト826およびネットワークランシーバオブジェクト816と同様に、その他の開発システムの構成要素に接続される。ARSオブジェクト828は、メッセージ制御オブジェクト826に接続され、そこから通告(D2Bメッセージ、送信結果、およびイベント)を受信する。処理の後に、ARSオブジェクトは、受信した全ての通告をデバイスオブジェクト830(以下参照)に伝達する。従って、ARSオブジェクトは、メッセージ制御オブジェクトおよびデバイスオブジェクトとの接続を確立する機能と、最終的にそれらの接続を破棄する機能を提供する。これらの機能は、専用のAPI中に設けられる。

#### 【0115】

ARSクラスは、現在、以下のことを行う4つのAPIを提供する。

- ・ARSの属性(例えばどのSDFを使用するか、テキスト翻訳を使用するかどうか、など)を設定すること。
- ・ARSをその他の開発システムの構成要素に接続すること。
- ・接続されたメッセージ制御オブジェクトから通告を受信すること。
- ・SDFのシステム仕様へのアクセスを提供すること。

#### 【0116】

(デバイス830)

デバイス構成要素は、構成要素マネージャであると考えることができ、システムインスタンスの中心となる構成要素である。システムインスタンスの概念については既に述べたが、分かりやすくするためにここでさらに詳細に述べる。

#### 【0117】

クライアントアプリケーション810が開発システムを使用するときには、システム制御オブジェクト832、デバイスオブジェクト830、ARSオブジェクト828、メッセージ制御オブジェクト826、および（任意選択で）スイッチボックスオブジェクト834が作成される。これらのオブジェクト（以下で詳述する）は、事実上アプリケーションに属し、ともにアプリケーションのシステムインスタンスを形成する。これらのオブジェクトはアプリケーション810に固有のものであり、その他のアプリケーションがアクセスすることはできない。これは、OPBオブジェクトやネットワークランシーバオブジェクト、コードックオブジェクト、ディレクタオブジェクト（後述）など、その他の開発システムの構成要素とは対照的である。これらは共用される構成要素であり、複数のアプリケーションがアクセスし、使用することができる。クライアントアプリケーションは、いくつかのAPIを介してその開発システムの属性および挙動を制御することができる。

#### 【0118】

クライアントアプリケーションは、OPBを制御してこの開発システムを使用するときには、ネットワーク中の特殊な状態およびアプリケーションプロトコルを有する、D2Bデバイスとして事実上動作する。このD2Bデバイスは、D2Bアドレス（その物理的なリング位置とは異なる）や、それがD2Bシステムマスタであるかどうかなどの属性を有する。開発システムのデバイス構成要素830は、クライアントアプリケーションがこれらの属性を設定し、取出すことができるようにするオブジェクトである。さらに、開発システムは全体として、例えば、それがD2Bメッセージのテキスト翻訳を実行するかどうか、またはスイッチボックス構成要素を使用するかどうかなどの属性を有する。第1の属性は、事実上ARS828の属性であり、第2の属性は、開発システム全体としての属性であるが、アプリケーションが使用している完全な開発システムについてのこの

情報をカプセル化するのはデバイス構成要素830である。これは、様々な属性を設定し、取出することができるようにするAPIを表明する。

#### 【0119】

クライアントアプリケーションは、開発システムを使用するときには、システム制御オブジェクト832を作成する。このオブジェクトが最初に行うのは、デバイスオブジェクトを作成することである。次いで、デバイスオブジェクト830が、開発システムのその他の構成要素（すなわちメッセージ制御、ARS、およびスイッチボックス）の作成を開始し、これらを互いに接続する。次いで、広域ディレクタオブジェクト824（必要なら作成する）にアクセスし、空いているOPBオブジェクト814を取出す。最後に、そのOPBのネットワークトランシーバ816にアクセスし、それ自体のメッセージ制御826とネットワークトランシーバオブジェクト816を互いにリンクする。これにより、OPBドライバ構成要素812からデバイスオブジェクト830までの、全体の経路がもたらされる。クライアントアプリケーションが終了したときに、デバイスオブジェクトは、それが接続する全ての構成要素のリンクを破棄し、そのシステムインスタンス中のオブジェクトを破壊する。

#### 【0120】

デバイスオブジェクト830は、最初にどのOPBオブジェクト（従ってどの物理的OPB）をクライアントアプリケーションのシステムインスタンスが使用するかを決定する。デバイス830は、クライアントが、OPBオブジェクト、そのトランシーバオブジェクト、およびコーデックオブジェクト（以下参照）にアクセスすることができるようにするAPIを表明する。しかし、PCに取り付けられたOPBは複数存在することもあり、エンドユーザが特定のOPBを選択して使用したいと望むこともある。クライアントアプリケーションは、ディレクタ構成要素824を介して、PCに接続されたOPBを指図し、1つを選択し、そのシステムインスタンスにそれを使用するよう命令することができる。デバイスのAPIがこれを行うための機能を提供し、デバイスオブジェクトは、新たに指定されたハードウェア構成要素を、それ自体のシステム構成要素への接続を行う。

## 【0121】

デバイスオブジェクトは、その他の構成要素と同様に、接続されたハードウェア構成要素（802、804）からクライアントアプリケーション810に、イベントを渡す役割を果たさなければならない。デバイスオブジェクトは、自動応答サーバ（ARS）オブジェクト832を作成して、デバイスオブジェクト自体に接続し、それによりARSがイベントをメッセージ制御826からデバイスオブジェクト自体に渡すことができるようにする。ただし、デバイスオブジェクトは一意的であり、その他の構成要素がそれら自体をデバイスオブジェクトに接続し、そこから通告を受信することができるようにするために、明確に定義された標準APIを表明する。図8から分かるように、デバイスは、システム制御オブジェクト832およびスイッチボックスオブジェクト834の両方に接続される。システム制御オブジェクトは、デバイスオブジェクトを作成した後、標準APIを使用してそれと結合する。同様に、デバイスオブジェクトがスイッチボックスオブジェクトを作成した後、スイッチボックスオブジェクトは同じAPIを使用して、それ自体をデバイスオブジェクトに結合する。その後、デバイスオブジェクトが受信した全ての通告は、自動的にシステム制御オブジェクトおよびスイッチボックスオブジェクトに伝達される。

## 【0122】

この標準機構は、任意の構成要素が、デバイスオブジェクトと結合するために使用することができる。例えば、特定のデバイス構成要素と結合することによってクライアントアプリケーションを「スパイ」することができるGUIアプリケーションを構築することもできる。これを行おうとする構成要素には、以下の2つの条件がある。

- ・この構成要素は、デバイスオブジェクトがその構成要素に通告を渡すための特定のAPIを実装しなければならない。
- ・この構成要素は最初に、デバイスオブジェクトに接続するためにそのデバイスオブジェクトへの何らかの参照を有さなければならない。

## 【0123】

これは、そのデバイスオブジェクトへの参照を既に有しているその他の開発シ

システムの構成要素が、その参照を供給し、または外部エージェント（ディレクタ構成要素824など）がそれを供給することを意味する（これは、ディレクタが実行中の全てのデバイスオブジェクトを認識しているようにするための機構が、存在しなければならないことになることを意味する）。

#### 【0124】

（システム制御832）

システム制御オブジェクト832は、クライアントの、開発システム環境とのインタフェースである。開発システムを使用したいと望むあらゆるクライアントアプリケーション810は、そのアプリケーション中でシステム制御オブジェクトをインスタンス化しなければならない。そして、このオブジェクトが、開発システムを制御するためのそのアプリケーションの「ブリッジ」またはインタフェースとして働く。システム制御オブジェクト832は、クライアントアプリケーションに対して単一のAPIを表明し、またシステム制御オブジェクトが、そのアプリケーションに通告を渡すことができるようにするための、定義されたAPIを表明するようそのアプリケーションに求める。

#### 【0125】

システム制御オブジェクト832が作成されると、デバイスオブジェクト830をインスタンス化し、これがその後、対応するアプリケーション810に特有の開発システムの残りの部分の構築を開始する（上記ではシステムインスタンスと呼ぶ）。システム制御オブジェクト832自体は、機能性を実質上含まず、単にその全てのAPI呼出しを接続されたデバイスオブジェクトに委任するだけである。システム制御オブジェクトの主な仕事は、アプリケーションからAPI機能呼出し中に渡されたデータを、開発システム内部で使用される適当なフォーマットに翻訳すること、およびその逆の翻訳をすることである。

#### 【0126】

システム制御オブジェクト832は、デバイスオブジェクト830を作成した後は、そのデバイスの標準機構を使用してそれと結合し、これによりシステム制御オブジェクトは、デバイスオブジェクトから通告を受信することができる。従って、システム制御オブジェクトは以下の2つのインタフェースを表明する。

- ・開発システムを制御するアプリケーション810のためのクライアントアプリケーションインタフェース。
- ・システム制御オブジェクトにイベントを通告するためのデバイスオブジェクトによって使用される通告インタフェース。

#### 【0127】

クライアントアプリケーション810が最終的に終了したときに、システム制御オブジェクト832は、そのデバイスオブジェクト830との接続を切断し、次いでデバイスオブジェクトを破棄する。デバイスオブジェクトは、それ自体が破棄される間に残りのシステムインスタンスを破棄し、「クリーンアップ」する。

#### 【0128】

単一のアプリケーションが、1以上のシステム制御オブジェクト832をインスタンス化することも可能である。この場合には、作成されるそれぞれが、その他のシステム制御オブジェクトのインスタンスとは別個の、それ自体の完全に一意的なシステムインスタンスを構築することになる。異なるシステム制御オブジェクトはそれぞれ、別個のインタフェースボード(OPB814)を制御するために使用することができる。

#### 【0129】

(スイッチボックス834)

D2B Optical技術の最も重要な利点の1つは、単一の光ファイバを介して複数チャネルのソースデータを伝達できることである。ソースデータの送信を管理制御するために、D2BソースデータプロトコルおよびD2Bスイッチボックスサブデバイスが規定されている。あらゆるD2Bデバイスは、スイッチボックスおよびそれと関連するソースデータプロトコルをサポートするためのソフトウェアを実装しなければならない。これは、この開発システムを介してOPBを制御する任意のクライアントアプリケーションも、D2B Opticalネットワーク中で相互動作する「開発システムデバイス」(図9の電話エミュレーション109'など)のための、このソフトウェアを実装しなければならないことを意味する。いうまでもなく、これは全ての開発システムのアプリケーション

ンにとって望ましくない負担である。

#### 【0130】

この問題を多少とも解消するために、この開発システムは、スイッチボックスクラスのオブジェクトを有する。スイッチボックスオブジェクト834は、開発システムを使用するクライアントアプリケーションの代わりに、D2Bスイッチボックスおよびソースデータ接続プロトコル論理を効果的にカプセル化し、実装する。D2Bメッセージを受信すると、デバイスオブジェクト830はこれをスイッチボックスオブジェクト834に渡し、次いで、そのメッセージがスイッチボックスメッセージであるかどうかを調べる。スイッチボックスメッセージである場合には、スイッチボックスオブジェクト834はこのメッセージを処理し、必要な任意の動作を実行する、すなわち新しいスイッチボックスメッセージを光ネットワーク上に伝達し、それ自体のネットワークトランシーバにあるソースデータの経路指定を制御する。

#### 【0131】

デバイスオブジェクトは、スイッチボックスオブジェクトを作成したときに、システムインスタンスが使用しているネットワークトランシーバオブジェクト816へのアクセスをスイッチボックスに与える。そして、スイッチボックスオブジェクトは、D2Bスイッチボックスメッセージに応答して、トランシーバの経路指定情報テーブル(routing information table)(RIT)または等価な経路指定制御を自由に改変することができる。クライアントアプリケーション810が、そのシステムインスタンスに異なるOPBを使用するよう命令した場合には、デバイスオブジェクト830は、スイッチボックスオブジェクトが常に正しいネットワークトランシーバオブジェクトへのアクセスを備え、それを操作することを保証する。

#### 【0132】

様々なトランシーバICのRITは、様々な(ポート数、ポートサイズ、構成)であり、スイッチボックスは様々なトランシーバタイプのいずれにも接続することができるので、スイッチボックスオブジェクトは各トランシーバの型についての特有の知識をカプセル化する。理想的には、スイッチボックスは一般的な挙動

のみを含むべきであり、各ネットワークトランシーバクラスは、スイッチボックスがそのトランシーバのRITに関する特定の知識にアクセスできるようにするAPIを表明すべきである。その場合には、スイッチボックスオブジェクトは、この知識を動的に使用して、そのトランシーバにとって正しいRIT操作を実行するはずである。しかし、現在のトランシーバおよびスイッチボックスの設計はこの考えをサポートしていない。

### 【0133】

クライアントアプリケーション810は、図4に示すように、現実のデバイスをエミュレートするために、ソースデータを光ネットワーク上に、または光ネットワーク外で経路指定しようとすることがある。このソースデータは、トランシーバIC上の特定のポートを介して経路指定されることになり、論理的にD2Bサブデバイスと関連することになる。スイッチボックスオブジェクト834は、この情報自体を決定することができ、従って、クライアントアプリケーションからこれを通知される。この目的のために、スイッチボックスオブジェクトは、それを構成することができるようにするインタフェースを表明する。ユーザは、デバイスオブジェクト830を介してこのAPIにアクセスすることができる。このAPIにより、ユーザは、どのD2Bサブデバイスが論理的にトランシーバICに接続されるかを、それらが接続されるトランシーバ上の実際の入出力ポートと同様に指定することができる。スイッチボックスオブジェクトは、ARSオブジェクトを使用してSDFのシステム仕様にアクセスし、特定のサブデバイスと関連するソースデータタイプを決定する。このように、スイッチボックスオブジェクトは、トランシーバのポートおよびそのポートと関連するデータのフォーマットについての知識を有するので、ソースデータをサブデバイスから、またはサブデバイスに経路指定する必要があるときに、どのRITチャンネルを操作すべきかを知っている。

### 【0134】

ソースデータ接続の構築で、スイッチボックスが使用されるときには、スイッチボックスは、クライアントアプリケーションに通告を送信することによって通知する。これらの通告は、接続が設定／破棄されたかどうか、その接続のID、

およびその接続の影響を受けるサブデバイスなどの情報を含む。さらに、その接続と関連するチャンネル情報（すなわちソースデータフィールド（図5）内のどのチャンネルをその接続が使用しているか）も与えられる。スイッチボックスオブジェクト834は、これらの通告を、それが接続されたデバイスオブジェクト830に渡す。デバイスオブジェクト830は、通告を受信するために、既にARSオブジェクト828に対してAPIを表明している。スイッチボックスオブジェクトはこれと全く同じAPIを使用する。

#### 【0135】

スイッチボックス自体は以下の3つのインタフェースを表明する。

- ・スイッチボックスを構成するためのクライアントインタフェース。
- ・初期化とスイッチボックスオブジェクトをネットワークトランシーバオブジェクトに接続するために、デバイスオブジェクトが使用する制御インタフェース。
- ・デバイスオブジェクトが通告をスイッチボックスに渡すための通告インタフェース。

#### 【0136】

ここまでは、クライアントアプリケーション810が常にスイッチボックス構成要素のサービスを使用することを望むものと仮定している。しかし、ユーザがそれ自体のスイッチボックスの論理および動作を実装することを望むこともあることから、そのようにならないこともある。これをサポートするために、その主要なAPI中のデバイスオブジェクト830は、スイッチボックス構成要素の使用を切替えるための機能を設ける。この属性がオフである場合には、デバイスオブジェクトはスイッチボックス構成要素を作成しない（それが存在すれば破棄する）。その後、属性がオンに切り替わった場合に、スイッチボックスオブジェクトを作成することになる。オフに切り替わった場合には、クライアントアプリケーションは、トランシーバのレジスタを操作し、D2Bメッセージを適切に取り扱うことによって、そのアプリケーション中でスイッチボックスの挙動を実装する役割を全面的に担う。

#### 【0137】

（デバイスデータベース836）

プロトコルおよび機能が、ある範囲の特定の製品が事前に配列したフォーマットで符号化されることによって、実際に実装するところのデバイスデータベース836を設けることもできる。ユーザがエミュレートのための専用アプリケーションをプログラミングすることなく、デバイスの記述である「ライブラリ」を、ARSオブジェクト828とともに使用して、選択された製品を開発システム内でエミュレートすることができる。製造業者は、このデバイスの記述であるライブラリを共用し、新製品を開発するにつれて拡張し、互換性のある製品の開発を大きく促進することができる。

### 【0138】

#### (クライアントイベント)

システムインスタンスは、クライアントアプリケーションからのコマンドに回答する単なる受動システムではない。システムインスタンスは、任意の時間に起こるイベントを、システム制御オブジェクト832を介して非同期にクライアントに通知することができる。これらのイベントは、外部のD2B Opticalネットワーク上、インタフェースファームウェア822中、およびシステムインスタンス自体の中で起こった事態から、発生させることができる。この例では、システムインスタンス中のイベントを、以下のカテゴリに一般化することができる。

- ・受信されたD2Bメッセージ。
- ・クライアントアプリケーションから送信されたメッセージについてのD2Bメッセージの送信結果。
- ・ネットワーク管理状態の変化およびその変化の原因。多くの様々なイベントがこのカテゴリに入る。多くの様々な事柄が、状態の変化、例えばロックはずれ／確立、タイムアウト、ネットワークエラーなどを引き起こすからである。
- ・ネットワークトランシーバのレジスタの書込み。これらは、任意のネットワークトランシーバのレジスタがうまく書き込まれたときに発生する。
- ・内部エラー。例えば、クライアントアプリケーションが、何らかのシステムインスタンスのプロパティに無効値を設定しようとした場合、または内部システムのフレームワークにエラーが発生した場合など。

## 【0139】

以下の表は、システムがクライアントアプリケーションに信号を送ることができるイベントの一部を定義したものである。

## 【0140】

【表1】

イベントタイプ	記述
メッセージ受信	クライアントがアプリケーションプロトコルメッセージを受信したときに発生する。このメッセージはこのイベントとともに渡される。
光学的起動	OPB上の光学的起動トリガが、掛かったときに発生する。
電氣的起動	OPB上の電氣的起動トリガが、掛かったときに発生する。
遠隔スタートアップ	スタートアップの後に、光学的リングが作動状態になった時に発生する。
遠隔遮断	光学的リングが遮断された時に発生する。
ロックはずれ	クライアントのネットワークトランシーバで、ネットワークのビットストリームのロックがはずれたときに発生する。
ロック確立	クライアントのネットワークトランシーバが、確立したときに発生する。
障害報告	ネットワーク管理障害、例えばデバイスの喪失が報告されたときに発生する。

## 【0141】

【表2】

リング中のノード	マスタが、スタートアップの後に、リング中に新しいデバイスを検出するたびに発生する。ノードの数はこのイベントで渡される。
スタートアップ失敗	ネットワークが正しくスタートアップできないときに発生する (PCT/GB98/ (62796WO) 参照)。
2相/パリティエラー	ネットワークトランシーバが、ネットワークのビットストリーム中で、2相符号則またはパリティエラーを検出したときに発生する。
OPB通信エラー	OPBドライバと、クライアントがネットワークトランシーバを制御しているインタフェースボードとの間の通信が、失われたときに発生する。
ノード位置	ネットワークトランシーバが、スタートアップの後に、そのノード位置を確立した時に発生する。ノード位置はこのイベントで渡される (EP-A-0725516 (95P03) 参照)。
メッセージ送信OK	ネットワーク上のアプリケーションプロトコルメッセージの送信が成功するたびに発生する。実際のメッセージまたは識別子は、このイベントとともに渡され、従ってクライアントは、このイベントがどのメッセージと関係するかを知ることができる。

【0142】

【表3】

メッセージ失敗	「メッセージ送信OK」であっても、メッセージが送信できないときに発生する。
R I T変更	スイッチボックスが、ソースデータの経路指定のためにネットワークトランシーバのR I Tを改変したときに発生する。R I Tレジスタおよびその新しい値は、このイベントで渡される。このイベントと関連する接続IDを渡すこともできる。
アドレス初期化	ネットワークトランシーバが、ネットワークのスタートアップが成功した後で、そのデバイスアドレスを確立した時に発生する。このアドレスはこのイベントで渡される (PCT/GB98/00872 (62792WO) 参照)。
警報モード	ファームウェアが、警報モードに入ったときに発生する。
障害回復モード	OPBファームウェアが、障害回復モードに入ったときに発生する。
受光	ネットワークトランシーバが、電氣的起動が掛かった後で受光したときに発生する。
ロックタイプアウト	デバイスが、電氣的起動が掛かってから、設定期間内にロックが確立しない場合に発生する。
位置報告タイムアウト	デバイスが、スタートアップ時にロックが確立した後の設定期間内に、ReportPosition状態要求を受信しない場合に発生する。
遮断タイムアウト	マスタデバイスが、遮断後の設定期間内に、全てのデバイスからPower Off状態報告を受信しないときに発生する。

## (開発システムの初期化例)

図11は、初期化中の開発システムの、様々な構成要素モジュール間の動作および通信のシーケンスを示している。各構成要素の動作は、それぞれの時間線上に表し、図8と同じ参照符で表示してある。

## 【0144】

ステップ1100で、クライアントアプリケーション810は、運用を開始するために、システム制御オブジェクト832のインスタンスを作成する。1102で、システム制御オブジェクトが、デバイスオブジェクト830を作成し、1104でデバイスオブジェクトが、ディレクタオブジェクト824を作成する。

## 【0145】

1106で、ディレクタオブジェクト824が、OPBオブジェクト814を作成する。1108から1114で、OPBオブジェクトは、ネットワークランシーバオブジェクト816、コーデックオブジェクト818を作成し、インタフェースボードのファームウェアエレメント822およびネットワーク管理エレメント820を初期化する。

## 【0146】

1116および1118で、デバイスオブジェクト830はさらに、ARSオブジェクト828およびメッセージ制御826をそれぞれ作成する。1120で、ARSオブジェクト828は、対応するメッセージ制御オブジェクト826とのリンクを確立する。

## 【0147】

1124で、デバイスオブジェクト830は、コマンドGetComponent (“OPB”, “0”) をディレクタ824に送信し、1126で、ディレクタは、予約されたOPB814の識別をこのアプリケーションに戻す。1128で、デバイスオブジェクト830は、OPB814にそれが空いていることを確認するよう要求し、ステップ1130で確認を受信する。1132で、デバイスオブジェクト830は、開発システムのこのインスタンスの運用の準備ができていることをシステム制御832に通知する。

## 【0148】

## (開発システムの動作例)

図12は、ネットワーク上へのD2Bメッセージの送信を示している。1200で、クライアントアプリケーションは、システム制御832に対して機能呼出しを行い、送信したいと望む、「CDチェンジャ」にアドレス指定された「play」「FORWARD」などのD2Bメッセージを、テキスト表現で提供する。1202で、システム制御832は、この呼出しをデバイスオブジェクト830に渡し、デバイスオブジェクトが1204でこれをARSオブジェクト828に渡す。ARSオブジェクト828は、設定されている警告レベルに従って、またシステム記述ファイル838中のプロトコルおよびシステム記述に従ってこのメッセージを検査する。このメッセージは受諾または拒絶される。

## 【0149】

エラーが拒絶された場合には、1206で、ARS828はデバイスオブジェクト830にエラーコードを送信し、1208でデバイスオブジェクトは同じエラーコードをシステム制御オブジェクトに渡す。1210で、関連するエラーコードを含むイベントが、クライアントアプリケーション810に通信される。

## 【0150】

メッセージが警告レベルおよびシステム記述ファイルに従うテストに合格したと仮定すると、ARS828はテキストフォーマットのメッセージを16進数に翻訳し、1212でこれをメッセージ制御826に渡す。上記のメッセージの例について16進数フォーマットは、0x1c8、0x190、0x0e、0x42、0x90、0xc3、0x75となり、ここで、最初の2つのコードは送信元アドレスおよび送信先アドレスである。1214で、メッセージ制御オブジェクト826は、メッセージをネットワークランシーバオブジェクト816に渡し、1216でネットワークランシーバオブジェクトがこれをOPBオブジェクト814に渡す。ここで、OPBオブジェクト814がOPBドライバを使用してメッセージをインタフェースボードおよびD2B Opticalネットワークに送信する。

## 【0151】

1218で、送信結果が、ネットワークランシーバオブジェクトからメッセ

ージ制御オブジェクト826に戻される。ここから、送信結果は、ステップ1220から1224で、ARS828およびデバイスオブジェクト830を介して、システム制御832に渡される。最後に、1226で、システム制御オブジェクト832が、伝送情報および完全なメッセージの表現を含むイベントをクライアントアプリケーション810に送信する。

#### 【0152】

当業者ならば、本発明の趣旨および範囲内で、上記実施形態の多くの変更形態が可能であることを容易に理解するであろう。さらに、記載した開発システムおよびその重要な任意の機能を、D2B Opticalシステム以外のプロトコルで使用するために、またConan製品の範囲外のトランシーバ構成要素に合わせて、適合させることもできる。

#### 【図面の簡単な説明】

次に、添付の図面に関連して、例示のみを目的として本発明の実施形態について述べる。

#### 【図1】

様々なデバイスが環状ネットワーク中で通信する範囲を含むローカル通信システムを示すブロック概略図である。

#### 【図2】

図1のシステム中で使用される、制御およびソースデータアーキテクチャを示す図である。

#### 【図3】

統合インタフェースを備えたステーションを示す図である。

#### 【図4】

図1のインタフェースモジュールの1つを示す概略図である。

#### 【図5】

既知のD2B Opticalフォーマットに従って伝送される、デジタル信号のフレーム構造を示す図である。

#### 【図6】

図1のシステム中の装置間で伝送される、デジタル信号のフレーム構造を示す

図である。

【図7】

図6のフレーム構造内で伝達される制御フレームの構造を示す図である。

【図8】

図1のネットワークと互換性のある装置を開発する際に使用される、製品開発およびテストツールの、ソフトウェアおよびハードウェアエレメントを示す図である。

【図9】

開発中の製品のエミュレーションにおいて、開発ツールの使用を示す図である。

【図10】

図1のネットワークとの互換性を検証するために、いくつかのデバイスをテストする際に、製品開発ツールの使用を示す図である。

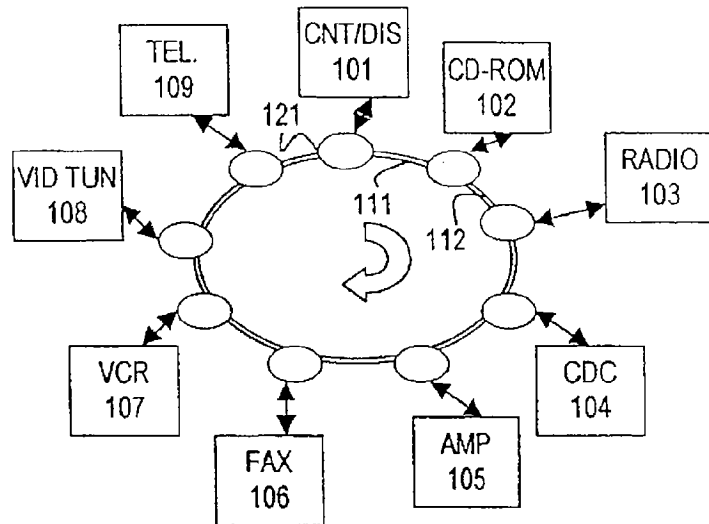
【図11】

開発システムを初期化する際の様々な構成要素モジュールの動作を示す図である。

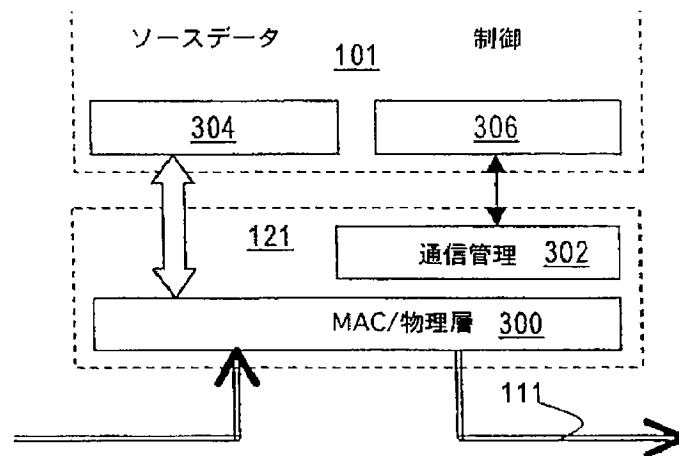
【図12】

製品のエミュレーションの一部として、メッセージを送信する際の構成要素モジュールの動作を示す図である。

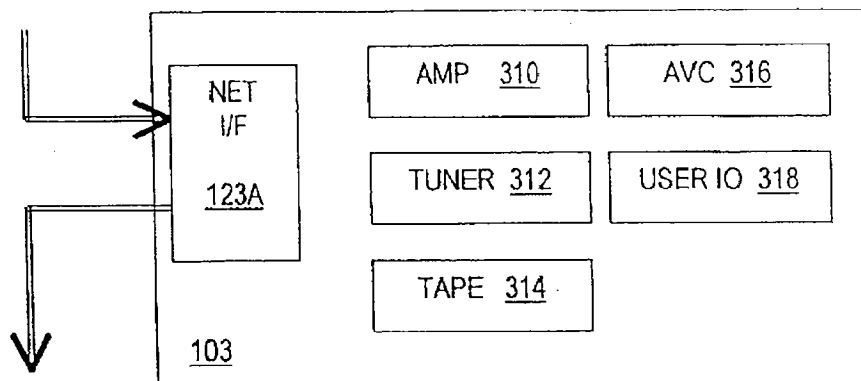
【図1】



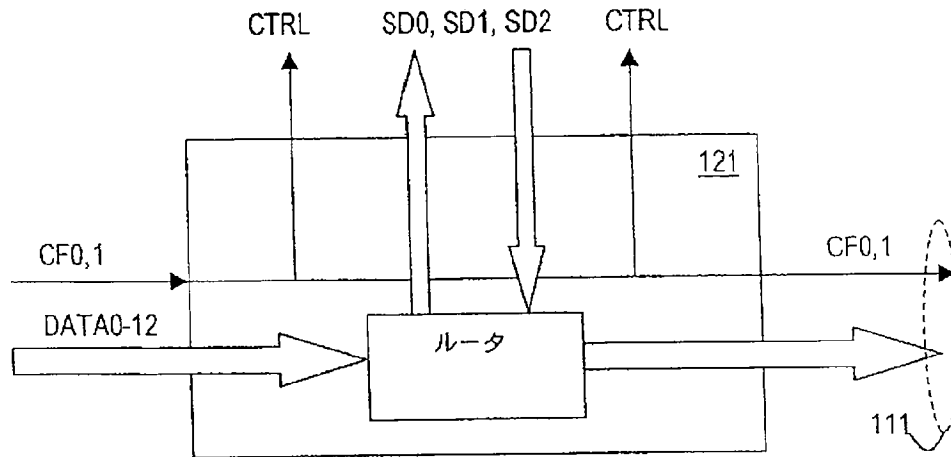
【図2】



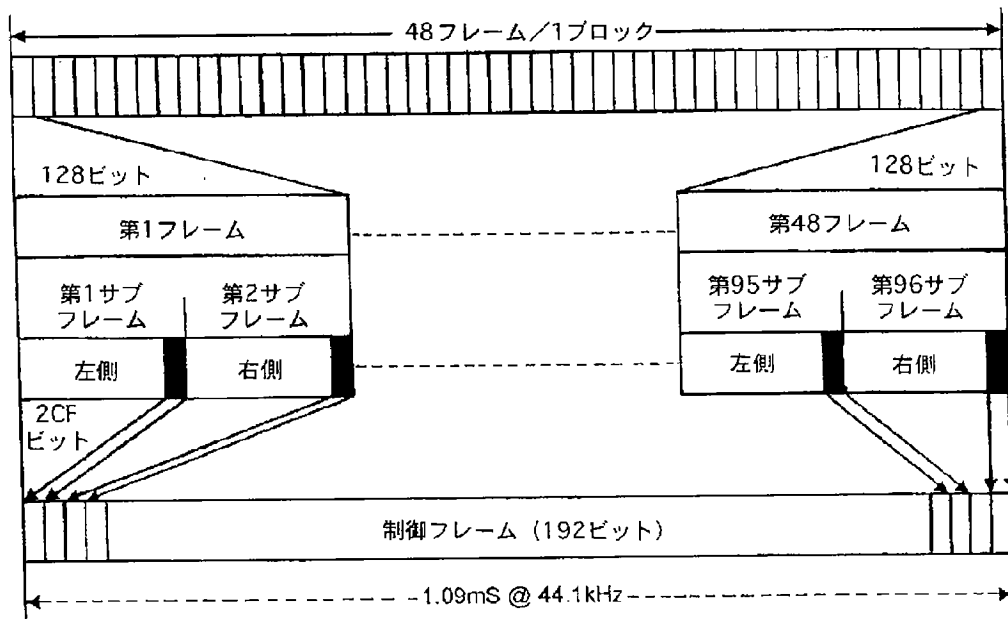
【図3】



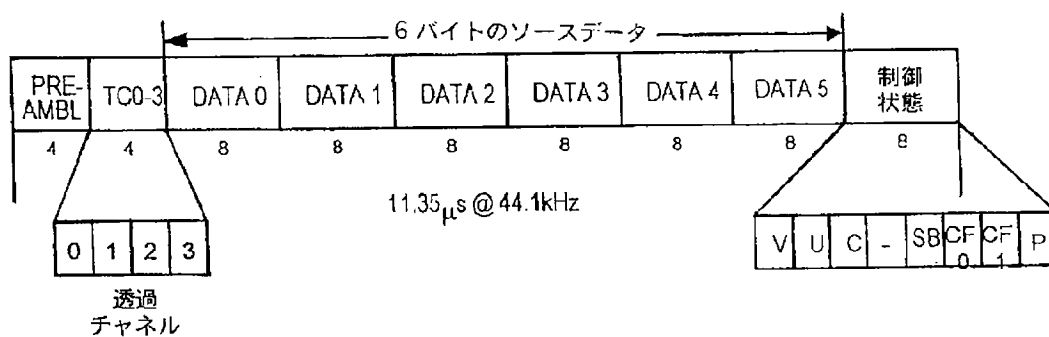
【図4】



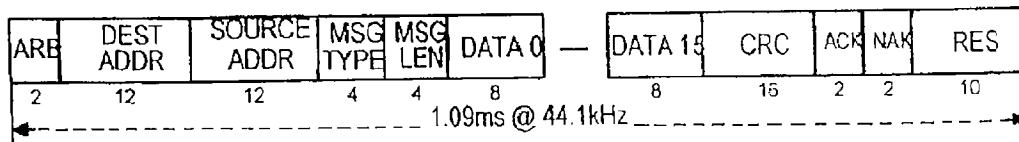
【図5】



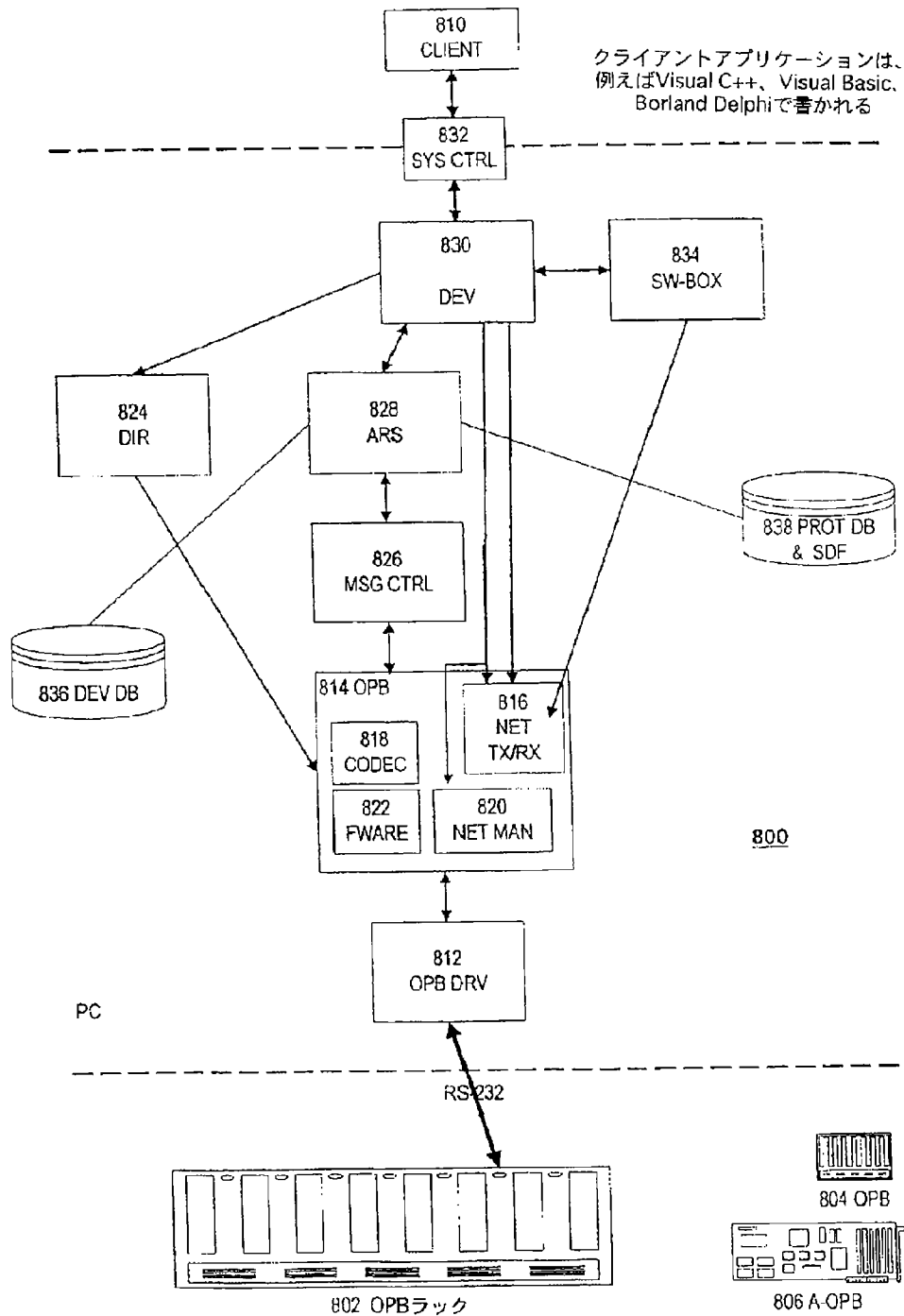
【図6】



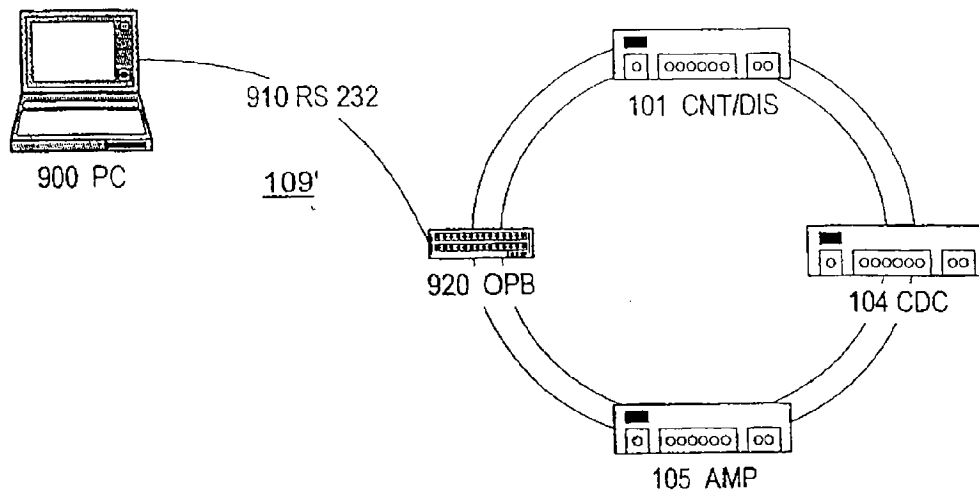
【図7】



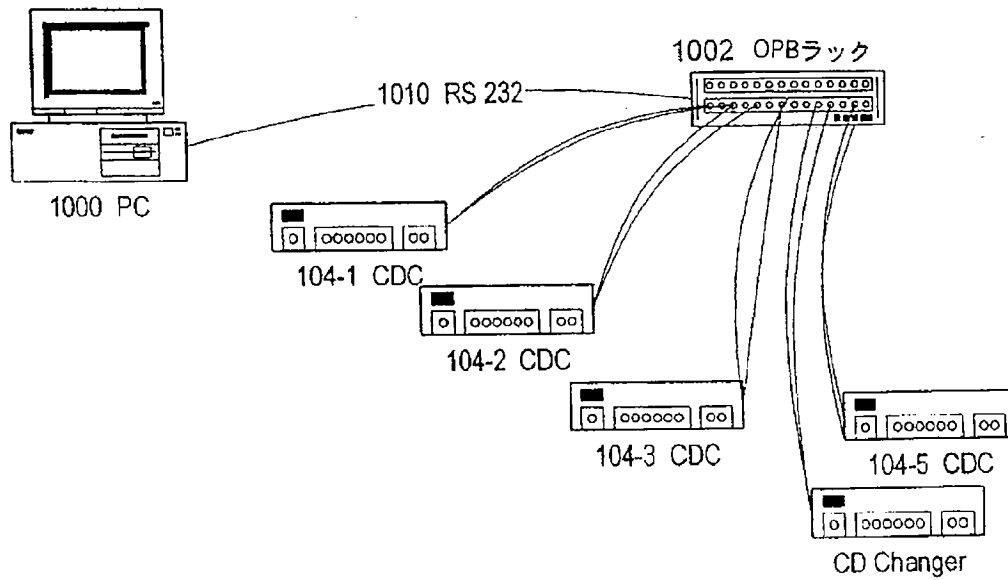
【図8】

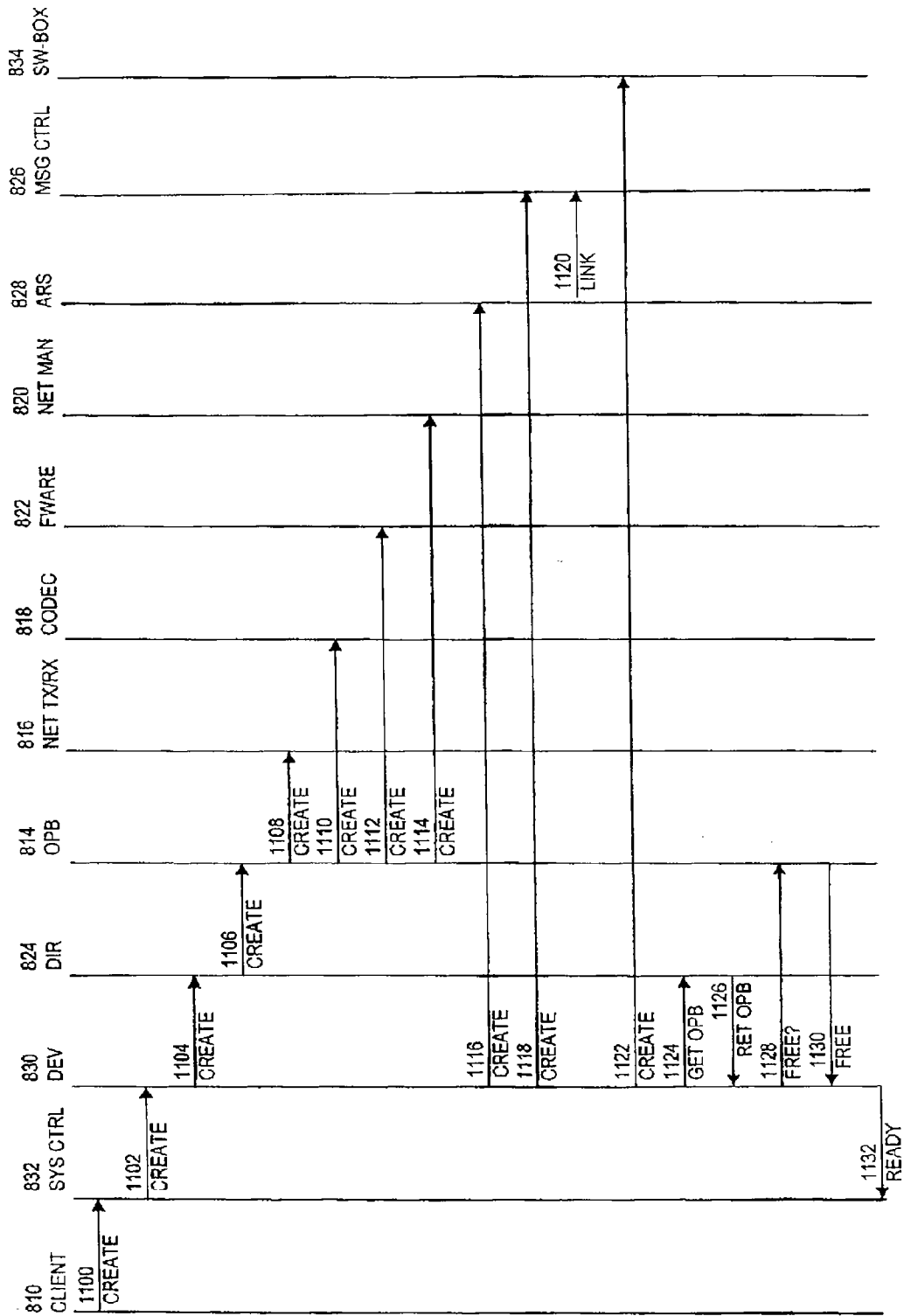


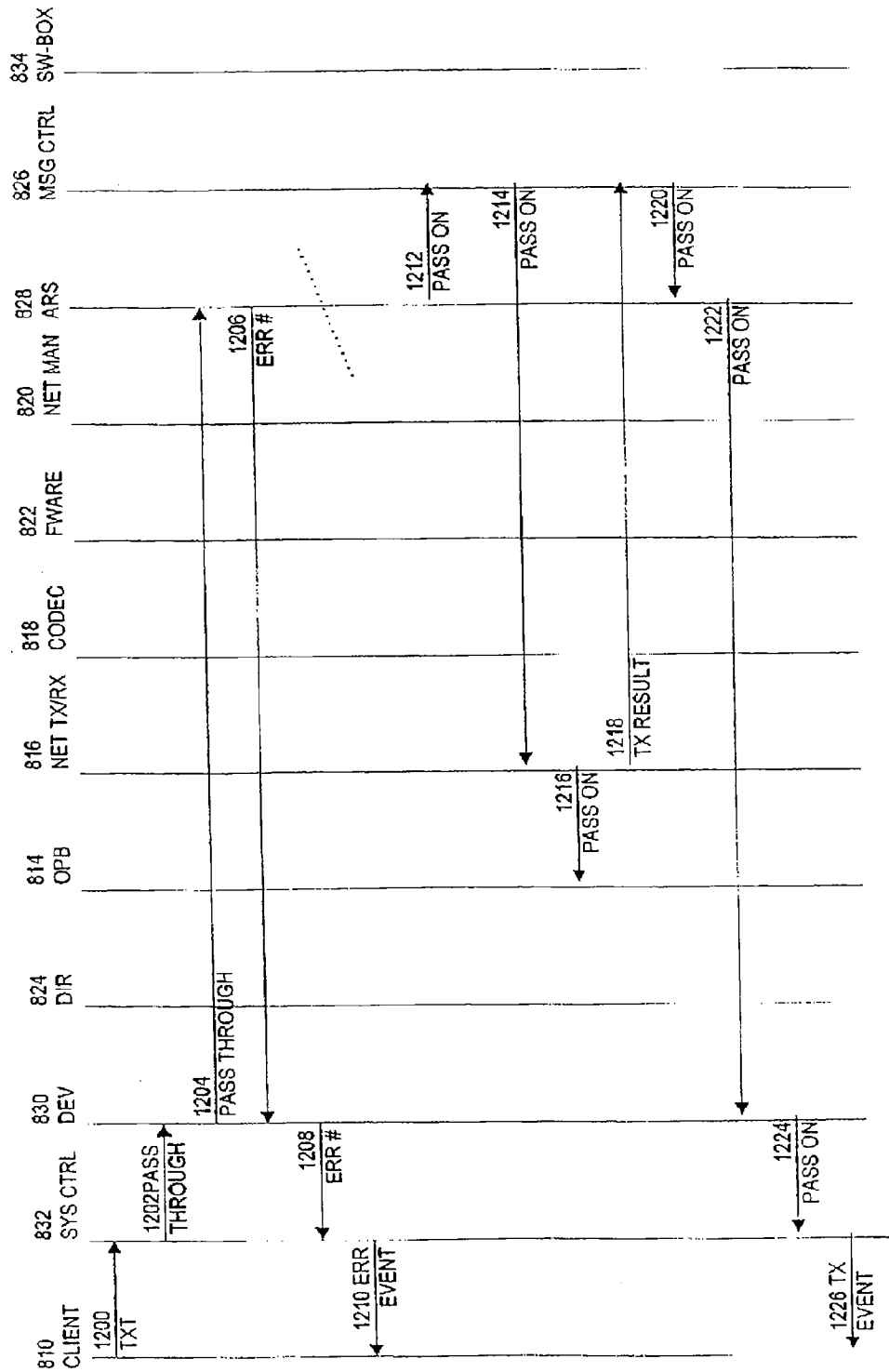
【図9】



【図10】



【☒ 1 1】

【☒ 1 2】

## 【国際調査報告】

INTERNATIONAL SEARCH REPORT		International Application No. PCT/GB 98/02721
A. CLASSIFICATION OF SUBJECT MATTER IPC 6 H04L12/26 H04L29/06		
According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) IPC 6 H04L H04Q H04M		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched		
Electronic data base consulted during the international search (name of data base and, where practical, search terms used)		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 97 23076 A (N B NETWORKS) 26 June 1997 see figures 1-16 see page 1 - page 41	1-3, 5, 7-14
A	EP 0 788 267 A (SUN MICROSYSTEMS INC) 6 August 1997 see figures 5-9 see page 1 - page 7	1-14
A	VRANKEN H P E ET AL: "DESIGN FOR TESTABILITY IN HARDWARE-SOFTWARE SYSTEMS" IEEE DESIGN & TEST OF COMPUTERS, vol. 13, no. 3, 21 September 1996, pages 79-87, XP000627796 see the whole document	1-14
-/-		
<input checked="" type="checkbox"/> Further documents are listed in the continuation of box C. <input checked="" type="checkbox"/> Patent family members are listed in annex.		
* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "E" earlier document but published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document relating to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art. "G" document member of the same patent family		
Date of the actual completion of the international search  12 May 1999		Date of mailing of the international search report  25/05/1999
Name and mailing address of the ISA European Patent Office, P.O. 5818 Patentkan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016		Authorized officer  Eraso Helguera, J

## INTERNATIONAL SEARCH REPORT

International Application No.  
PCT/GB 98/02721

C. (Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT		
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	TARNAY K ET AL: "PROCONSUL, A TOOL FOR COMPUTER-AIDED PROTOCOL ENGINEERING" MICROPROCESSING AND MICROPROGRAMMING, vol. 38, no. 1 / 05, 1 September 1993, pages 821-825, XP000383845 see the whole document -----	1-14
A	BUDKOWSKI S: "ESTELLE DEVELOPMENT TOOLSET (EDT)*" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 25, no. 1, 3 August 1992, pages 63-82, XP000294189 see the whole document -----	1-3

## INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/GB 98/02721

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9723076 A	26-06-1997	US 5793954 A AU 1355097 A EP 0868799 A US 5781729 A	11-08-1998 14-07-1997 07-10-1998 14-07-1998
EP 0788267 A	06-08-1997	US 5822520 A JP 9326796 A	13-10-1998 16-12-1997

## フロントページの続き

(81)指定国 EP(AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), DE, GB, JP, US

(71)出願人 Stirling House, Stirling Road The Surrey Research Park Guildford, Surrey GU2 5RF England

(72)発明者 スクミドット マークス ジョアンヌズ  
イギリス ケイティー6 4ビーエックス  
サリー サービスン グローブ ロード 15

Fターム(参考) 5B014 EA06 EB03 FA05 FB03 GD05  
GD22 GD23 GD49 HC05  
5B042 GA12 GB09 GC08 GC12 HH07  
HH12  
5B089 GA21 JB03 JB05 JB07 KA12  
KB09 KC15 KC28 KC47 KF04  
MC15  
5K034 CC02 CC05 DD03 KK27 TT02  
5K035 BB03 EE13 HH02 HH07

## 【要約の続き】

ケーションプロトコルに加えて、ツールは、正常コマンドの範囲をさらに限定するために、ネットワーク上に存在するまたはエミュレートされるデバイスの特定の数および型を規定するネットワークシステムの記述を記憶する。